JAN & MARCH 1978 KIM-1/6502 USER NOTES ISSUES 10 & 11

HAVE YOU BEEN ON THE BUS?

The 'Neatest Development of the Year' award has got to go to the COMMODORE PET computer for its use of the IEEE 488 (GPIB) General Purpose Interface Bus for all communication I/O. Although the bus is somewhat difficult to understand, at first, the real advantage of utilizing this method of I/O handling becomes apparent when you consider that only one piece of interface hardware and one software driver routine can handle up to 15 different devices at varying data transfer rates.

This clearly indicates what we can expect in future 'personal' computers.as it fits in so neatly with the concept of distributed intelligence in system design.

I feel certain that other equipment manufacturers will follow suit and adopt this bus into new gear, but, in any case, it will be quite intersting to see what develops in this area.

Has anyone interfaced KIM to the IEEE Bus? Would you be interested in a tutorial article on the basic concepts of the bus? If I can find the time, I'll try to get something together for the next issue.

ERIC

A FLOPPY DISC FOR KIM................................(finally)

-the editor-

I used to dream of the day when I'd be able to hook KIM up to a floppy disc! Now, at work anyway, my dream has come true!!!

A company called HDE in New Jersey has interfaced KIM to a SYKES disc/controller combination and has written some neat software to make the whole thing work together like a system, not like a bunch of parts thrown together.

The operating system is file oriented (like some high-class mag-tape systems you've probably heard about) and includes a version of the MOS assembler/editor as an integral part. Assembly language programmers will really appreciate the ability to work with named object and source files. The ability to load a 6K source file in less than a half a second really made it clear what a time saver this system could be. (Without the disc, it works out to about one-third to one-fourth time being wasted just waiting for slow tape being read or written to)

The Editor has actually been spruced up a bit from its original form and makes the system quite easy to operate as well as being quite powerful in function.

FODS, as it's called requires the top 8K of RAM for its storage, and is bootstrapped in via a short program that is easily loaded in via tape.

For more info contact:HDE, box 120, Allamuchy, NJ 07820 (phone 201-852-9268) or Johnson Computer, box 523, Medina, Ohio (phone 216-725-4560)

KIM-1 USER NOTES IS PUBLISHED BI-MONTHLY (whenever possible) by Eric C. Rehnke, 109 Centre Ave., West Norriton, PA 19401. Subscription rates are $5.00 for six issues (U.S. & Canada) and$10.00 elsewhere. No part of the USER NOTES may be copied for commercial purposes without the expressed written permission of the publisher. Articles herein may be reprinted by club newsletters as long as proper credit is given and the publisher is provided with a copy of the publication.

HARDWARE REVIEW

BY THE EDITOR

MEMORY-PLUS FROM THE COMPUTERIST

Sooner or later, the question of memory expansion enters the minds of most KIM users. Here's another alternative from the same folks who brought us PLEASE (a play package), HELP (a work package), and MICRO (a newsletter dedicated entirely to machines of the 6502 genre).

The thing that really interested me was the way this board was configured. Besides having an 8K block of RAM, MEMORY-PLUS includes sockets for 8K of Intel 2716 EPROM, a complete programming facility for the 2716, and the MOS Technology 6522 VIA (Versatile Interface Adaptor). I prefer to call it the VVIA (VERY VERSARTILE INTERFACE ADAPTOR). I'm sure you'd agree after studying the 24 page spec sheet that accompanies this device.

But back to MEMORY-PLUS....

The built-in 2716 programmer requires the user to supply +25 volts, but this can be gotten easily from three 9 volt transistor batteries hooked up in series. The programming software is, of course, included as is a memory test program and a 60 page manual.

Since MEMORY-PLUS is the same size and shape as KIM, it can be mounted directly beneath the KIM by means of 1" stand-offs. Hardware was provided for this purpose, but it proved unsatisfactory so suitable stand-offs were found elsewhere. Rubber feet are included to protect the bottom of the board and an optional set of pre-wired connectors is available to speed up assembly time. By the way, MEMORY-PLUS comes fully assembled, tested and includes a 90 day warranty, (just like KIM). All IC's are socketed and battery backup of the RAM is provided for, if needed.

It's really quite impressive to have all this power in so small a package. The next step is to get an assembler/editor and extended I/O monitor "burned" into a few 2716's and turn this two-board machine loose as a low-cost development system.

About the only negative comment I can make about MEMORY-PLUS is that further memory expansion could be slightly difficult. Definitely not just a matter of plugging in another board. This may not be a disadvantage in certain applications, but should be considered.

MEMORY-PLUS costs $245.00 and is available from: The COMPUTERIST, P.O. Box 3, S. Chelmsford, Ma. 01824 617-256-3649. Get their catalog of other KIM products.

The fol[illegible]ng program utilizes the now famous driver circuit on page 57 of the Kim U[illegible]s Manual. Although it is set up to provide the sound of four phasar bursts, it can easily be modified in a number of ways to provide all kinds of neat sounding effects.

Location 201 sets no. of repeats (00 to FF).

Location 207 in conjunction with 209 set the length of tone before increment/decrement 207 (00 to FF); 209 (04 to 07).

One interesting variation is to load: 203 with FF
21d with c6 (dec)
222 with 00

Among other sounds you should be able to make a "Bomb Drop Whistle" and a "Red Alert" condition.

The program is relocatable and uses one page zero location (EE). The program could also easily be converted to a subroutine leaving you no excuse for not adding sound effects to your next program.

R2D2-Eat your heart out!

```
200   A0   04          LDY #04
202   A9   00          LDY #00
204   85   EE          STA  EE
206   A9   01       4  LDA #01
208   8D   06   17     STA 1706
20B   A9   01          LDA #01
20D   8D   01   17     STA 1701
210   EE   00   17  2  INC 1700
213   A6   EE          LDX (EE)
215   CA            1  DEX
216   D0   FD          BNE 1
218   2C   07   17     BIT 1707
21B   10   F3          BPL 2
21D   E6   EE          INC (EE)
21F   A5   EE          LDA EE
221   C9   FF          CMP #FF
223   F0   02          BEQ 3
225   D0   DF          BNE 4
227   88            3  DEY
228   F0   02          BEQ 5
22A   D0   DA          BNE 4
22C   4C   4F   1C  5  JMP 1C4F
```

EDITOR'S NOTE; I've been having great fun with this routine. All kinds of sounds are possible and the program can be easily integrated into most any game program-see Butterfield's SKEET SHOOT program elsewhere in this issue.

••••••••••

SKEET SHOOT   September/77   Jim Butterfield, Toronto

Start the program and you'll see targets racing across the screen from right to left. You don't have to fire at any of them .. but if you do, remember that you must 'lead off' your shot to give the bullet time to reach the target. You have 20 shots: shoot by hitting any numbered button. You'll see the bullet move from right to left, too. If you hit the target, you'll see the explosion. After 20 shots, KIM will tell you the number of hits you made; then you can press GO for another game.

```
0200 A2 00     START  LDX #0       reset hit counts
0202 86 F9            STX HITS
0204 86 FA            STX POINTL
0206 86 FB            STX POINTH
0208 A9 13            LDA #$13     19+1 shots
020A 85 D0            STA SHOTS
020C CA               DEX          set X=$FF
020D 86 D1            STX BULLET   ..no bullet, and
020F 86 D2            STX TARGET   ..no target
0211 A5 D2     MAIN   LDA TARGET   is there a target?
0213 10 0D            BPL FLIGHT   yes, continue
0215 AD 04 17         LDA TIMER    no, make random target
0218 29 3F            AND #$3F     not too slow..
021A 09 0C            ORA #$0C     ..and not too fast
021E 29 0B            AND #$0B     place off screen
0220 85 D2            STA TARGET   ..in random position
0222 C6 D4     FLIGHT DEC TARSPD   count down delay
0224 D0 06            BNE SIGHT    time to move target?
0226 A5 D3            LDA SPEED    yes, restore count down
0228 85 D4            STA TARSPD
022A C6 D2            DEC TARGET   move the target
022C A5 D1     SIGHT  LDA BULLET   is bullet in flight?
022E 30 06            BMI CLEA     no, skip bullet move
0230 C6 D5            DEC BULSPD   count down delay
0232 D0 06            BNE CLEAR    time to move bullet?
0234 C6 D1            DEC BULLET   yes, move it
0236 A9 08     CLEA   LDA #$8       reset..
0238 85 D5            STA BULSPD      ..countdown
023A D8        CLEAR  CLD
023B 20 40 1F         JSR KEYIN    directional registrs
023E 20 6A 1F         JSR GETKEY   test keyboard
0241 C5 D6            CMP LAST     same key?
0243 F0 12            BEQ TRIG     yes, skip key action
0245 85 D6            STA LAST     keep new key ID
0247 C9 10            CMP #$10     numeric key?
0249 B0 0C            BCS TRIG     no, skip key action
024B A5 D1            LDA BULLET   bullet already in flite?
024D 10 08            BPL TRIG     yes, don't fire
024F A2 06            LDX #6       position bullet right
0251 86 D1            STX BULLET
0253 86 D7            STX STRIKE   no hit yet
0255 C6 D0            DEC SHOTS    one less shot left
0257 A9 7F     TRIG   LDA #$7F     set dirct regstrs
0259 8D 41 17         STA PADD
025C A2 05            LDX #5       show six digits
025E A0 13            LDY #$13     set digit #6
0260 A9 00     LITE   LDA #0       start with digit blank
0262 E4 D1            CPX BULLET   ..if bullet in this spot
0264 D0 03            BNE NOBUL
0266 BD B0 02         LDA BTAB,X   ..put in in segment
0269 E4 D2     NOBUL  CPX TARGET   ..if target in this spot
026B D0 02            BNE NOTARG
026D 49 21            EOR #$21     add target segments
026F C9 20     NOTARG CMP #$20     a hit?
0271 D0 10            BNE SHINE    no, skip ahead
0273 A5 D7            LDA STRIKE   have we counted it?
0275 30 0C            BMI SHINE    yes, skip
0277 F8 18            SED CLC      no, count it
0279 A5 F9            LDA HITS
027B 69 01            ADC #1       .. in decimal
027D 85 F9            STA HITS
027F A9 FF            LDA #$FF     explosion display
0281 85 D7            STA STRIKE   .. set counted flag
0283 8D 40 17  SHINE  STA SAD
0286 8C 42 17         STY SBD
0289 C6 D8     ZAP    DEC ZIP
028B D0 FC            BNE ZAP
028D 88 88 CA         DEY DEY DEX
0290 10 CE            BPL LITE     more digits?
0292 C9 FF            CMP #$FF     explosion?
0294 D0 04            BNE ENTES    no, skip next
0296 A5 D4            LDA TARSPD   delay..
0298 85 D5            STA BULSPD   ..display
029A A5 D1     ENTES  LDA BULLET   shot complete, and..
029C 25 D0            AND SHOTS    ..last shot?
029E 30 03            BMI QUIT     yes, show score
02A0 4C 11 02         JMP MAIN     no, keep going
02A3 20 1F 1F  QUIT   JSR SCANDS   show score;
02A6 20 6A 1F         JSR GETKEY   test keyboard for
02A9 C9 13            CMP #$13     ..GO key
02AB D0 F6            BNE QUIT     if not keep going
02AD 4C 00 02         JMP START    if GO start over

02B0 01 40 08  BTAB   .BYTE 1,$40,8,8,8,8
02B3 08 08 08

02B6 end
```

"KIM D-BUG" by Lew Edwards

Want to eliminate the job of replacing an opcode with a BRK instruction, looking at each register separately, doing a conversion on the "P" register to find out which flags are set and how to change them, then restoring the opcode and setting a new break in place? "KIM D-BUG" can eliminate all that hassle for you! It lets you see the X, Y, & ACC registers at a single glance and select the one you want to alter with the stroke of a single key. Another keystoke shows all the flags in binary form, and permits toggling individual flags with the keys A thru F. You can jump from "KIM D-BUG" to KIM monitor and back at your pleasure, with full access to all monitor functions. "KIM D-BUG" automatically inserts the BRK opcode and the restores the original opcode when the break has executed, making a simple operation of the whole business.

To use "KIM D-BUG", start at 0100 and press "GO". Nothing happened? The IRQ and NMI vectors have been changed to the ones "KIM D-BUG" needs and you are now back in the monitor. Put your starting address into 00EF & 00F0 (low order first as usual), press "PC" and verify that this address is now in the program counter. Press "ST" and you will see KIM substitute 00 for the opcode at that address, then restore the original. You are now in the "KIM D-BUG" mode and will have a new set of responses to the keys. Pres "DA" and you will see X register contents on the left, Y register contents in the center, and ACC register contents on the right. You may now alter the contents of the ACC register via the HEX keys. If you press "+" or "GO", the display will remain the same, but the HEX keys will now alter the Y or X register respectively. Press "PC" and the display will switch to 1's and 0's indication flag conditions in order from left to right C,Z,V,I,N,D. Keys A thru F will set or reset the flags in the same order.

OK, got your initial values keyed in? Now press "AD", which causes a switch to KIM's monitor. Key in the address you want the break to occur and press "ST". You will see your START address displayed briefly, and then your BREAK address. Your program has now run from the first location to the second. If you want to return from the monitor to "KIM D-BUG" instead, you simply press the "PC" key, then "ST". The START and STOP will be the same and your program will stop before it gets started.(KIM D-BUG runs from PCL,H to POINTL,H), but you would be in "KIM D-BUG" mode.

Let "KIM D-BUG" help you find those elusive BUGS------HAPPY HUNTING!

```
0100 A9 01     START   LDA #01       initialize interrupt vectors
0102 8D FB 17          STA NMIH
0105 8D FF 17          STA IRQH
0108 A9 15             LDA #15
010A 8D FA 17          STA NMIL
010D A9 34             LDA #32
010F 8D FE 17          STA IRQL
0112 4C 16 1C  NOGO    JMP NOSAV     jump to monitor here
0115 A5 F9     NMIGO   LDA INH       "ST" key starts things here
0117 F0 F9             BEQ NOGO      won't run with BRK opcode
0119 85 ED             STA CODE      save valid breakpoint opcode
011B A9 00             LDA #00
011D A8                TAY           no offset for index
011E 91 FA             STA POINT,Y   substitute BRK opcode
0120 85 EE             STA HOLD      delay count
0122 A5 EF             LDA PCL       move 'from' address to window
0124 85 FA             STA POINTL
0126 A5 F0             LDA PCH
0128 85 FB             STA POINTH
012A 20 19 1F  LOOK    JSR SCAND     show it and stall a bit
012D C6 EE             DEC HOLD
012F D0 F9             BNE LOOK
0131 4C C8 1D          JMP GOEXEC    then run program
0134 85 F3     IRQGO   STA ACC       BREAK TIME!
0136 68                PLA           save the registers in standard
0137 85 F1             STA PREG      locations just like KIM
0139 68                PLA
013A 85 EF             STA PCL
013C 68                PLA
013D 85 F0             STA PCH
013F 84 F4             STY YREG
0141 86 F5             STX XREG
0143 BA                TSX
0144 86 F2             STX SPUSER
0146 A0 02             LDY #02
0148 A5 EF     BAK2    LDA PCL       back up PC 2 counts
014A D0 02             BNE NOPAGE    skip next if not page border
014C C6 F0             DEC PCH
014E C6 EF     NOPAGE  DEC PCL
0150 88                DEY
0151 D0 F5             BNE BAK2
0153 A5 ED             LDA CODE      put opcode back where it belongs
0155 91 EF             STA PCL,IY
0157 A5 EF     STOP    LDA PCL       transfer PC address to POINTER
0159 85 FA             STA POINTL
015B A5 F0             LDA PCH
015D 85 FB             STA POINTH
015F D8                CLD           binary mode for keys
0160 20 19 1F          JSR SCAND     show break address
0163 20 6A 1F          JSR GETKEY    & get keyboard input
0166 C9 14             CMP #14       PC key?
0168 F0 2B             BEQ FLAGS     yes, show flags
016A B0 EB             BCS STOP      too high, try again
016C C9 10             CMP #10       AD key?
016E F0 A2             BEQ NOGO      KIM takes over
0170 90 E1             BCC STOP      hex, try again
0172 85 FD             STA INDEX     use DA, + or GO as index value
0174 A2 03     MOVE    LDX #03
0176 B5 F2     MOVLP   LDA REG,X     move X, Y, & ACC registers
0178 95 F8             STA POINT,X   to window
017A CA                DEX
017B D0 F9             BNE MOVLP
017D 20 BF 01          JSR PUSH      show 'em & get a key
0180 C9 10             CMP #10       not a hex key?
0182 B0 D3             BCS STOP      change mode
0184 A6 FD             LDX INDEX     which register?
0186 16 E2             ASL REG,X     update it
0188 16 E2             ASL REG,X
018A 16 E2             ASL REG,X
018C 16 E2             ASL REG,X     shift out the old
018E 15 E2             ORA REG,X     add in the new
0190 95 E2             STA REG,X
0192 38                SEC
0193 B0 DF             BCS MOVE      & put it in the window
0195 A5 F1     FLAGS   LDA PREG      load flags
0197 4A                LSR           shift C flag to carry
0198 29 67             AND #67       mask unwanted bits
019A 90 02             BCC BICON
019C 09 10             ORA #10       replace the carry flag in new location
019E A2 03     BICON   LDX #03
01A0 48        BILP    PHA           save accumulator
01A1 29 11             AND #11       2 flags at a time in binary
01A3 95 F8             STA POINT,X   stick 'em in the window
01A5 68                PLA           recover accumulator
01A6 4A                LSR A         next pair
01A7 CA                DEX
01A8 D0 F6             BNE BILP      til done
01AA 20 BF 01  LITE    JSR PUSH      show & key time
01AD C9 10             CMP #10       hex key?
01AF B0 A6             BCS STOP      no, change mode
01B1 C9 0A             CMP #0A       decimal?
01B3 90 F5             BCC LITE      keep trying
01B5 AA                TAX           alpha, use as index value
01B6 BD C3 01          LDA TABLE,X   bit to flip in PREG
01B9 45 F1             EOR PREG      flip it
01BB 85 F1             STA PREG
01BD B0 D6             BCS FLAGS     & to the window

               SUBROUTINE "PUSH"
01BF 20 1F 1F  PUSH    JSR SCANDS    key down?
01C2 D0 FB             BNE PUSH      wait
01C4 20 1F 1F  KEY     JSR SCANDS    next key?
01C7 F0 FB             BEQ KEY       no, keep looking
01C9 20 6A 1F          JSR GETKEY    yes, which one?
01CC 60                RTS           take it back

01CD 01 02 10  TABLE "BIT FLIPPERS"       reference address 01C3
01CD 01 02 40 04 80 08      BYTES
```

How about a graphics output device for KIM? Roy is also working on some games (LIFE, STAR TREK etc.) and an analog input circuit. NEAT!!! ...ERIC

__GRAPHICS INTERFACE__ *from...Roy Flacco, Drexel Univ., Physics Dept., 32nd & Chestnut Ave., Phila. PA 19104*

Here's the graphics interface I told you about. It has gone through a number of revisions (hence the delay in getting it to you) but I think it is worth it. The whole thing sets up with plenty of room on a 4x6 perfboard, hardly loads the KIM lines at all (everything is buffered), outputs to almost any standard oscilloscope, and costs well under $30.

Basically the interface accepts two 8-bit parallel words (one at a time from FAØ-FA7), latches them alternately into two 8-bit data buffers (U1,U2), converts them into two positive analog voltages (via U3,U4) which are directly proportional to the data words so that ØØhex= 0.0 volts, and FFhex= 2.56 volts, and presents these voltages for presentation as an X-Y point on a scope CRT.

PBØ is used to latch the data—a positive transition latches the data into the X buffer, a negative transition latches the data into the Y buffer. The best way to do this is initialize PBØ to a 1 and then alternately DEC and INC PBD. This latches Y, then X.

In order to avoid the slewing of the DACs from causing a smeared display, the trailing edge of the X strobe generated by U5 initiates an unblanking pulse which turns on the CRT beam for a time set by VR1. The rest of the time the beam is blanked (turned off) by the normally-high output of U6. This convention is dictated by the type of scope; some scopes have a Z-axis (intensity mod.) which works in reverse, namely a positive level turns the beam on. In this case, merely use the Qoutput of U6 instead of the $\overline{Q}$ as shown on the schematic.

If your scope is AC-coupled on the Z-axis you may have to make some minor changes in the blanking pulse in order to avoid hot spots where the beam sits for long periods of time. One such change would be to trigger U6 from Q1 the same as U5 (use one of the A inputs on the 74121) and use the pulse to blank the beam only during the latching process. This requires some experimentation and will also depend on how you write your software.

The heart of the circuit is of course the DACs, which are type ZN425E available from Ferranti Electric Inc., East Bethpage Rd., Plainview, NY 11803. They go for $8 each. Ferranti, incidentally, is a great company to deal with—excellent turn-around, very helpful, friendly people, and they make really fine parts. Anyway, the chip is a 16-pin DIP containing an R/2R resistor ladder, bipolar switches, a precision 2.56 volt reference, and an 8-bit counter (which we don't use in this case). The counter is used in ADC applications and for generating ramps and such. The biggest advantage to using this chip is that the output is already converted to a voltage, as opposed to most DACs which have a current output. This means the usual I/V op-amp converter may be eliminated. Also the inclusion of an on-chip reference makes it extremely easy to use. If you want a different full-scale output voltage you may either add an op-amp at the output, or more interestingly, you may apply an analog voltage at the input of the R/2R ladder instead of the internal reference. This allows you to effectively multiply your analog voltage by your digital word. The useful range of this external voltage is 0 to +3.0 volts. For more info write for the data sheet.

Also, because of the dual-function aspect of the chip, it should be possible to construct an ADC/DAC using only a few more parts than this output-only DAC. The applications to games and graphics-sketching are too numerous to list in detail, but for example, how about a throttle for the Lunar Lander, or a chase game displayed on the CRT? I'm going to design one using a joystick over the next few weeks after I get Life up and running using this present interface.

One last thing about the scope you use; if it has AC-coupling on either the vertical or horizontal channels you are in for a smeared display due to the tendency of the beam to travel back to the origin. This is difficult or impossible to correct short of rebuilding your amplifiers or getting a newer scope. If the Z-axis is AC-coupled or non-existent, take heart, though. I have successfully converted my Tektronix 317 to DC-coupled blanking using a high-voltage level-shifting circuit, and would gladly pass it along if anybody needs it, or help designing another.

As a demonstration of the graphics, I wrote (and include) a little program which produces some of the prettiest pictures you ever saw. It resides entirely in page zero and uses less than half the page. The first time you run it you'll see why I named it Starburst; depending on the mask at ØØ11 and the initial points at ØØ75 and ØØ76 you can get hundreds of different fascinating displays which spin, explode, flash, and otherwise dazzle.

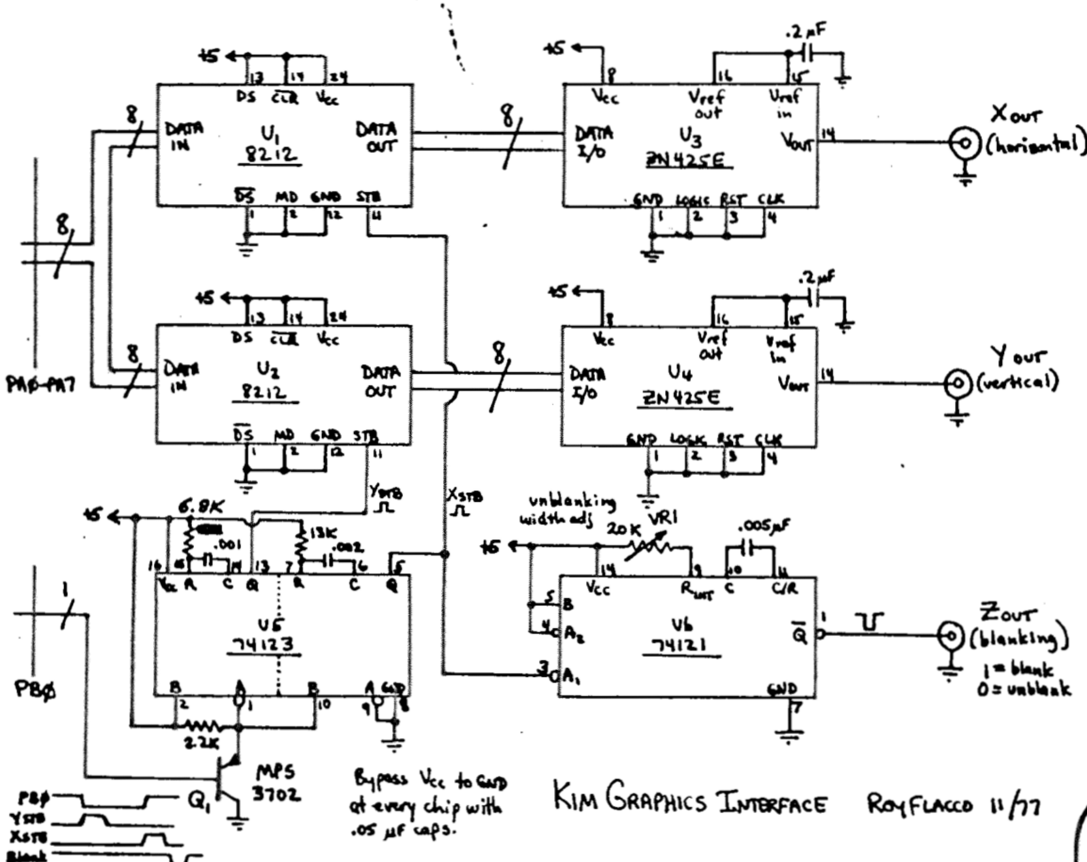
The use of an algorithm to generate the new point from the previous one exempts you from using much memory, since only a few coordinates are stored at any one time. The algorithm FULGEN is a variation on the ellipse-drawing one used in Aug. 77 BYTE, using 8-bit arithmetic. All overflow, underflow, and truncation errors are ignored, hence the rapidly moving display, which seems at times to bounce off the edges of the display screen and wrap around on itself. Using 16-bit arithmetic and taking care of over and under flow would help considerably toward stabilizing the picture, but frankly I like it more as it is.

HAFGEN calculates the proper coordinates for display in the four X-Y quadrants, since FULGEN works only on the first, and DISPLAY picks up the proper combination of halves and sends them to PROC which offsets them by 8Ø,8Ø to center the origin. I found it was necessary to include a DELAY loop between points to slow the motion down to a reasonable speed; changing this produces dramatic changes in the appearance. Note also that replacing the JMP at ØØ54 with the proper branch should make the program relocatable (there is a lot of flab in the program, like the LDX at ØØ43). I left it in to make it easier to see the program flow.

In writing your own software, bear in mind the basic format is LDA Ycoord./STA FAD/DEC PBD; then LDA Xcoord./STA PAD/INC PBD. Be sure to initialize PADD, PBDD, and PBØ at the start. Adjust RV1 for the brightest display without smearing.

STARBURST GRAPHICS

```
ØØ  A9 FF      START   LDA #$FF
Ø2  8D Ø1 17           STA PADD   Set PA for all outputs
Ø5  A9 Ø1              LDA #$Ø1
Ø7  8D Ø3 17           STA PBDD   Set PBØ for output
ØA  8D Ø2 17           STA PBD    Set PBØ=1
ØD  A5 76      FULGEN  LDA FULY   Generate new point FULX,FULY
ØF  4A                 LSR
1Ø  49 FE              EOR #$FE   try other masks; 7F, FD, etc.
12  38                 SEC
13  65 75              ADC FULX
15  85 75              STA FULX   new FULX
17  4A                 LSR
18  18                 CLC
19  65 76              ADC FULY
1B  85 76              STA FULY   new FULY
1D  4A         HAFGEN  LSR        scale-down into quadrants
1E  85 78              STA HFY    new half-Y
2Ø  49 FF              EOR #$FF
22  38                 SEC
23  69 ØØ              ADC #$ØØ
25  85 7A              STA NHY    new negative half-Y
27  A5 75              LDA FULX
29  4A                 LSR
2A  85 77              STA HFX    new half-X
2C  49 FF              EOR #$FF
```

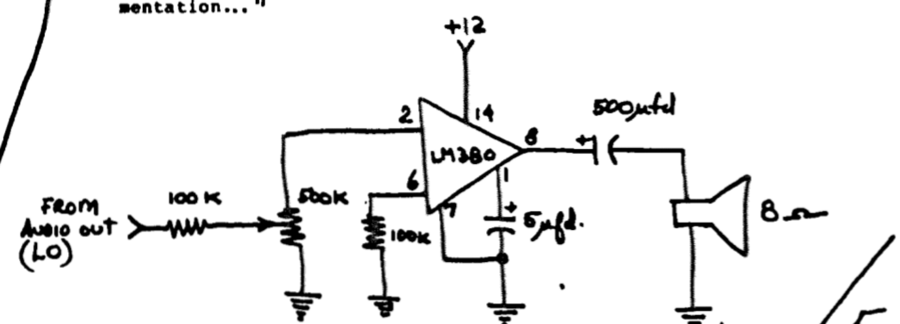


KIM GRAPHICS INTERFACE   ROY FLACCO 11/77

```
2E  38                  SEC
2F  69 ØØ               ADC #$ØØ
31  85 79               STA NHX      new negative half-X
33  AØ Ø1               LDY #$Ø1     number of display repeats
35  A6 77       DISPLAY LDX HFX
37  A5 78               LDA HFY
39  2Ø 57 ØØ            JSR PROC     Quadrant one: +X,+Y
3C  A6 79               LDX NHX
3E  A5 78               LDA HFY
4Ø  2Ø 57 ØØ            JSR PROC     Quadrant two: -X,+Y
43  A6 79               LDX NHX
45  A5 7A               LDA NHY
47  2Ø 57 ØØ            JSR PROC     Quadrant three: -X,-Y
4A  A6 77               LDX HFX
4C  A5 7A               LDA NHY
4E  2Ø 57 ØØ            JSR PROC     Quadrant four: +X,-Y
51  88                  DEY          Done displaying?
52  1Ø E1               BPL DISPLAY  No, do it again
54  4C ØD ØØ            JMP PULGEN   Yes, generate a new point
57  18          PROC    CLC          Processing and display
58  69 8Ø               ADC #$8Ø
5A  8D ØØ 17            STA PAD
5D  CE Ø2 17            DEC PBD      Latch Y-coord./blank CRT beam
6Ø  8A                  TXA
61  18                  CLC
62  69 8Ø               ADC #$8Ø
64  8D ØØ 17            STA PAD
67  EE Ø2 17            INC PBD      Latch X-coord./unblank CRT beam
6A  A9 Ø3       DELAY   LDA #$Ø3     Waste time between points
6C  8D Ø5 17            STA 8T       Load timer for 32 usec.
6F  2C Ø7 17    LOOP    BIT 1KT      Test for timer done
72  1Ø FB               BPL LOOP
74  6Ø                  RTS
```

Here's a circuit that looks useful to us cost-conscious KIM freaks from...James H. Van Ornum, 55 Cornell Dr., Hazlet, NJ 07730

"Finally, a circuit idea which is an aid for the cassette interface plus an output port for simple music programs. If you look at the audio tape interface schematic for KIM-1, you will notice PB7, audio out (hi) as well as audio out (lo) all have some form of the cassette signal during both record and playback. A high input impedance audio amplifier, using any of the audio IC chips readily available, provides a useful audio monitor during cassette IO as well as a single bit music port. The enclosed schematic provides the circuit details for my particular implementation..."



MORE ON TRIAC CIRCUITS (from Cass Lewart)

I checked again the waveforms of both my TRIAC interface circuits shown in issues 3 & 4 of the Newsletter and compared them with modifications suggested by Mike Firth and G. Thompson. I found the waveshapes and performance identical with that of my original circuits. In fact if one follows exactly Mike's suggestion to exchange MT1 and MT2 then the circuit will not work at all (Gate has to go to MT2 in either case). To answer Mike's question why I connect G to a point beyond the load, it is to obtain a better switching action as the gate voltage is then not affected by the variable load resistance. E.g. resistance of a 100W incandescent lamp varies from 10 Ohm when cold to 120 Ohm when hot. Though Mike doubts it (however, without checking), the circuit works fine and will not damage a motor. As the old saying goes: there are many ways to skin a cat!

RP[illegible] LCULATOR INTERFACE TO KIM

from...James Wood, 58 Hilltop Park, State College, PA 16801

In the last couple of issues of the KIM-1/6502 User Notes, Eric has mentioned the MM57109 "Number Cruncher Unit" (NCU) manufactured by National, and has noted that it should be easy, from a hardware and software standpoint, to interface to the KIM-1. Well, for those with the chip and the curious, here are the schematics and software listings of the interface that I am currently using to get the NCU and KIM-1 to parlé with each other. Also, I've included the details of my I/O expansion hardware (I've multiplexed peripheral port A) to complete the package of information.

Application I/O Interface

Hardware:

To start things out, we should first look at the Application I/O interface shown in Fig. 1. Peripheral port B is used by the interface to choose the appropriate input or output port. Below is the assignment of the bits of port B. Three bits are devoted

Ø	1	2	3	4	5	6	7
I	O	O	O	O	O	N/A	I
Sense Input	Aux Output	Port Select			#	N/A	IRQ

#used as a keyboard request signal in my system

to port selection; thus, you can potentially have up to 8 ports. In practice only 7 ports are used since the eighth port is used as a dummy I/O port (see below and subroutine OTSL). Typical input port and output port hardware are shown in Fig. 2. It should be noted that each port is either an input or an output not both, as one will find in an 8080 (8008) microprocessor system.

The two lower bits of port B are used as the input and output for the KIM-1 from and to, the sense inputs and auxiliary outputs respectively. The two multiplexed I/O bits were intended to serve as the handshake I/O lines, but their use is not limited to this application. One need only to remember that the two bits are inverted by the multiplexing chips and that the auxiliary outputs are normally low (active high). You will see that these two bits are extensively used by the NCU interface.

Software:

Three simple subroutines are all that you need to drive the Application I/O interface. They are INIT (Initialize data direction registers), INSL (Select an input port) and OTSL (Select an output port). I won't discuss the details of each subroutine, per se, since they are all well documented, except to state how they are used and a couple of precautions. To use OTSL and INSL, you just load the accumulator with the port # desired in bits 2(LSB), 3 and 4(MSB) with all other bits zero (bit 1 may be an exception), then jump to the appropriate subroutine. A word of caution: Never select an input port with OTSL, the results could be catastrophic since the 6530 outputs of the KIM-1 would be trying to drive the 74125 outputs. You should also be aware that port 7 should not be used since it is used by OTSL to allow a glitch free clearing of the chosen output port, i.e. no undefined states; consequently, the chosen output is always initialized to zero by OTSL.

After the mode (I or O) and port are selected, you need only execute a LDA 1700 or STA 1700 to complete the operation.

NCU Calculator Interface

Hardware:

The hardware that connects directly to the MM57109 is shown in Fig. 3.

There is nothing unique about this part of the interface since all the suggestions given by National in the NCU data sheets were followed. In brief, though, all outputs from the NCU are buffered with a 74LS367 gate with the appropriate pull-down resistor to V_{DD} on the gate's input. All TTL compatible inputs to the NCU have pull-up resistors to V_{SS} (V_{CC}). The clock has a frequency of approx. 400 KHz and uses a 74C04 run at 9V since the oscillator input as well as the HOLD and POR inputs are not TTL compatible.

The interface between the 74LS367's and the Application input buss is shown in Fig. 4. Again this interface follows closely the suggestions of National. Outputs DO1, DO2, DO3 and DO4 are latched into a 7475 by the $\overline{R/W}$ strobe which also sets a 7476 flip-flop. The BR output, if strobed, also will set a 7476 flip-flop. These flip-flops are reset by an auxiliary output signal from the Application interface after the KIM-1 has read the port. The ERR and RDY outputs of the NCU are also made available to the KIM-1.

The interface between the 74100 instruction latch and the Application output buss is shown in Fig. 5. This is a multi-purpose interface. Not only does it interface to the NCU circuitry, but it also interfaces with a "Beer Budget Graphics Interface" (BYTE, 1, 15, Nov., 1976). The circuitry for the latter is omitted but I shall explain the remaining circuitry pertinent to the NCU interface. Bits 06 and 07 are decoded to perform the instruction latching and hold function required in the NCU driving software. Briefly, 01XXXXXX (X=instruction bit) latches the instruction into the 74100, then 11XXXXXX brings the HOLD line low and the NCU commences the execution of the instruction. When the sense input #1 detects RDY=1 the KIM outputs 00XXXXXX and waits for RDY=0. More on this when we look at the driving software.

The last piece of hardware is the power supply. The NCU requires +5V and -4V. The +5V supply uses a 7805 and is self-explanatory. The -4V supply is derived from a -5V IC regulator whose output is further regulated to -3.9V with a zener diode. It should be noted that the capacitor of the size chosen on the output of the -5V regulator is necessary for the proper operation of the regulator.

This interface, as well as all the others, was constructed on Vector phenolic board. I used point-to-point wiring with a Vector wiring pencil. Sockets were used for the MM57109 and 74C04. The circuit worked the first time and has been running for about 6 months.

Software:

There are three basic subroutines which comprise the minimum needed to drive the NCU. They are CRST(Clear and reset NCU), EXEC (Execute a single word of an instruction) and OUTC(Get output from NCU). To fully utilize the capabilities of the NCU, you would need a jump, jump on condition, store and recall instruction subroutines, all of which would be similar in format to the OUTC subroutine. As it stands, the program MAIN allows you to write and execute a linear program (i.e. no jumps) and use only the registers in the NCU for storage.

To write a program for the NCU, you first write out the program in mnemonics, then translate the mnemonics into hexidecimal opcodes (See enclosed list of NCU opcodes). Then you load the encoded program into memory starting at 0300 (hex) up to a maximum of 255 steps. The last byte of the program must be FF to indicate to the KIM the end of the program. To start the program press AD 0200, the reset switch for the NCU, and then GO. After it is finished, the program will return to the KIM monitor and the output will be located in memory locations B0 to BC in one of two formats, described in the NCU data sheets, depending on whether the NCU is in scientific or floating point mode.

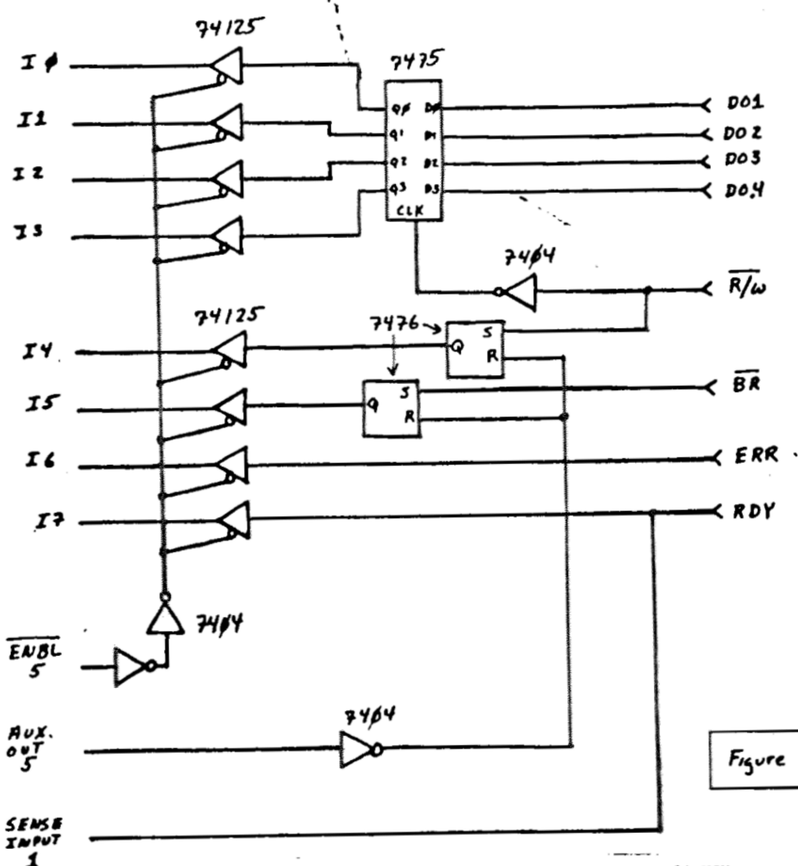


Figure 5

CALCULATOR CHIP SOFTWARE

```
0200 4C AF 02         JMP MAIN   Enter here if you want all NCU
                                 registers cleared
0203 4C B2 02         JMP CNTU   Enter here if you want registers
                                 undisturbed

0206 A9 00     INIT   LDA 00
0208 8D 01 17         STA 1701   Set data direction registers
020B A9 3E            LDA 3E     for Port A
020D 8D 03 17         STA 1703   and for Port B
0210 60               RTS

0211 48        INSL   PHA        Save port #
0212 20 06 02         JSR INIT   Set data direction registers
0215 68               PLA        Get port #
0216 8D 02 17         STA 1702   Select input port
0219 60               RTS

021A 48        OTSL   PHA        Save port #
021B 20 06 02         JSR INIT   Set data direction registers
021E A9 1E            LDA 1E     Select port 7
0220 8D 02 17         STA 1702
0223 A9 FF            LDA FF     Set Port A data direction
0225 8D 01 17         STA 1701   register for all bits as output
0228 A9 00            LDA 00
022A 8D 00 17         STA 1700   Clear Port A (all bits zero)
022D 68               PLA        Get output port #
022E 8D 02 17         STA 1702   Select output port
0231 60               RTS
```

```
0232 20 06 02  CRST   JSR INIT   Set data direction registers
0235 A2 05            LDX 05     Load accumulator with a NOP
0237 A9 3F            LDA 3F     instruction for NCU and
0239 20 54 02         JSR EXEC   do it 5 times so that
023C CA               DCX        NCU is now reset if reset
023D D0 F8            BNE CRST+5  switch was pressed.
023F A9 2F            LDA 2F
0241 20 54 02         JSR EXEC   Execute a MCLR instruction
0244 A9 14            LDA 14
0246 20 11 02         JSR INSL   Select port 5 (input)
0249 A9 16            LDA 16     Pulse Auxiliary output 5
024B 8D 02 17         STA 1702   to reset R/W and BR
024E A9 14            LDA 14     data latches
0250 8D 02 17         STA 1702
0253 60               RTS
0254 48        EXEC   PHA        Save instruction
0255 A9 04            LDA 04
0257 20 1A 02         JSR OTSL   Select port 1 (output)
025A AD 02 17  EXC1   LDA 1702   Check if
025D 4A               LSR A      RDY=1
025E B0 FA            BCS EXC1   (RDY=0)
0260 68               PLA        Get and
0261 48               PHA        Store instruction
0262 09 40            ORA 40     Put instruction in
0264 8D 00 17         STA 1700   instruction latch
0267 09 80            ORA 80
0269 8D 00 17         STA 1700   Set HOLD=0
026C AD 02 17  EXC2   LDA 1702   Check if
026F 4A               LSR A      RDY=0
0270 90 FA            BCC EXC2   (RDY=1)
0272 68               PLA
0273 8D 00 17         STA 1700   Set HOLD=1
0276 60               RTS

0277 A9 16     OUTC   LDA 16     Do an OUT instruction
0279 20 54 02  OUT1   JSR EXEC
027C 20 54 02         JSR EXEC   Second byte is ignored by NCU
027F A2 00            LDX 00     Initialize output buffer pointer
0281 A9 14            LDA 14
0283 20 11 02         JSR INSL   Select port 5 (input)
0286 2C 00 17  OUT2   BIT 1700   Check for no more data
0289 30 0F            BMI OUT3   (RDY=1)
028B AD 00 17         LDA 1700
028E 29 10            AND 10     Check for R/W flag set
0290 F0 F4            BEQ OUT2
0292 AD 00 17         LDA 1700
0295 29 0F            AND 0F     Load and
0297 95 B0            STA B0,X   Store digit
0299 E8               INX        Bump buffer pointer
029A A9 16     OUT3   LDA 16
029C 8D 02 17         STA 1702   Clear R/W Flag
029F A9 14            LDA 14
02A1 8D 02 17         STA 1702
02A4 2C 00 17         BIT 1700   Check if done (RDY=1)
02A7 10 DD            BPL OUT2
02A9 8A               TXA        Store buffer pointer
02AA 09 80            ORA 80     with bit 7 set to 1
02AC 95 B0            STA B0,X
02AE 60               RTS

02AF 20 32 02  MAIN   JSR CRST   Clear NCU registers
02B2 A0 00     CNTU   LDY 00     Initialize program pointer
02B4 B9 00 03  LOOP   LDA 0300,Y  Get instruction
02B7 C9 FF            CMP FF     Is it end of program?
02B9 F0 07            BEQ END    -if so, output # in NCU X register
02BB 20 54 02         JSR EXEC   -if not, do it
02BE C8               INY        Bump program pointer
02BF 4C B4 02         JMP LOOP   Do next instruction
02C2 20 77 02  END    JSR OUTC   Output X register of NCU
02C5 4C F4 1C         JMP MONITOR  Back to KIM
```

NCU CALCULATOR INTERFACE

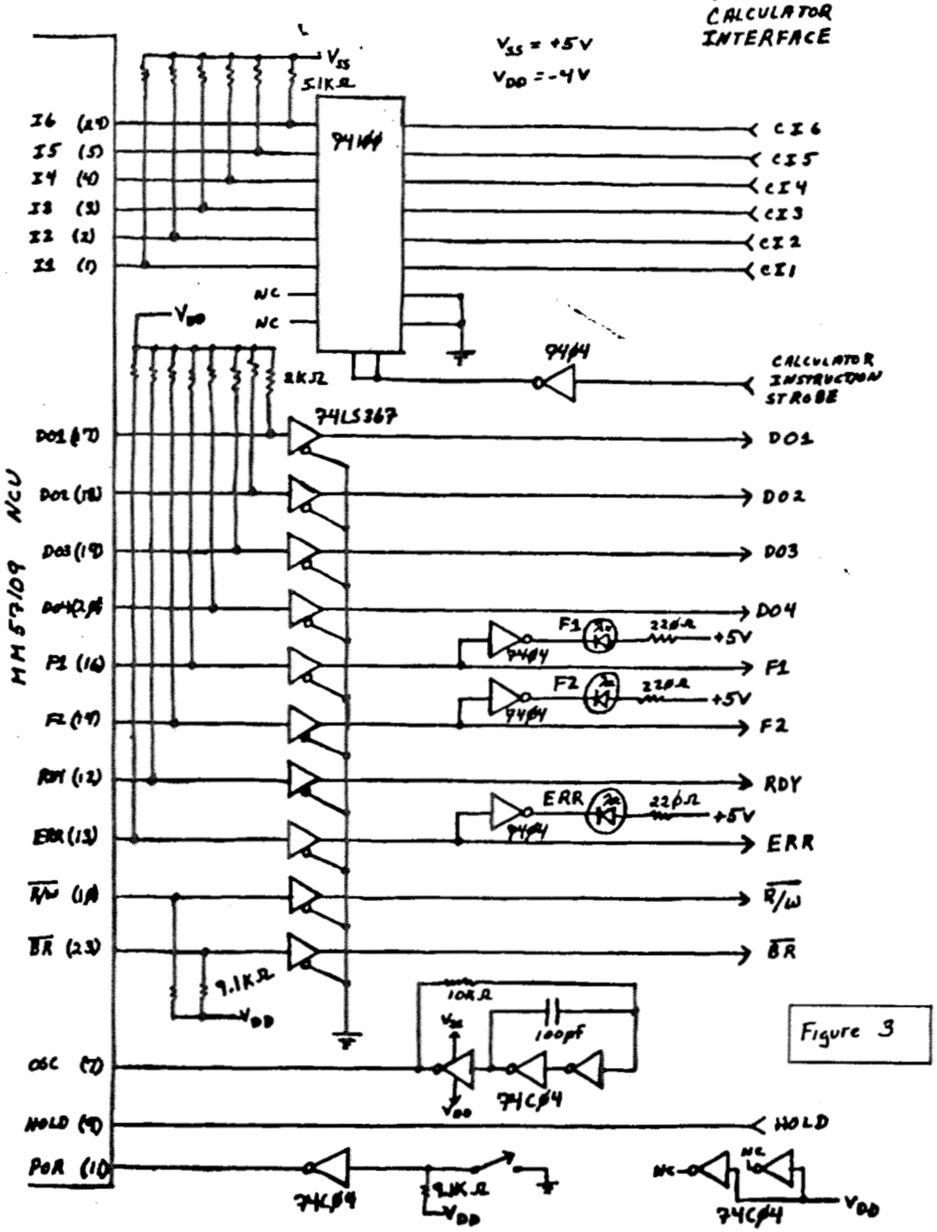


Figure 3

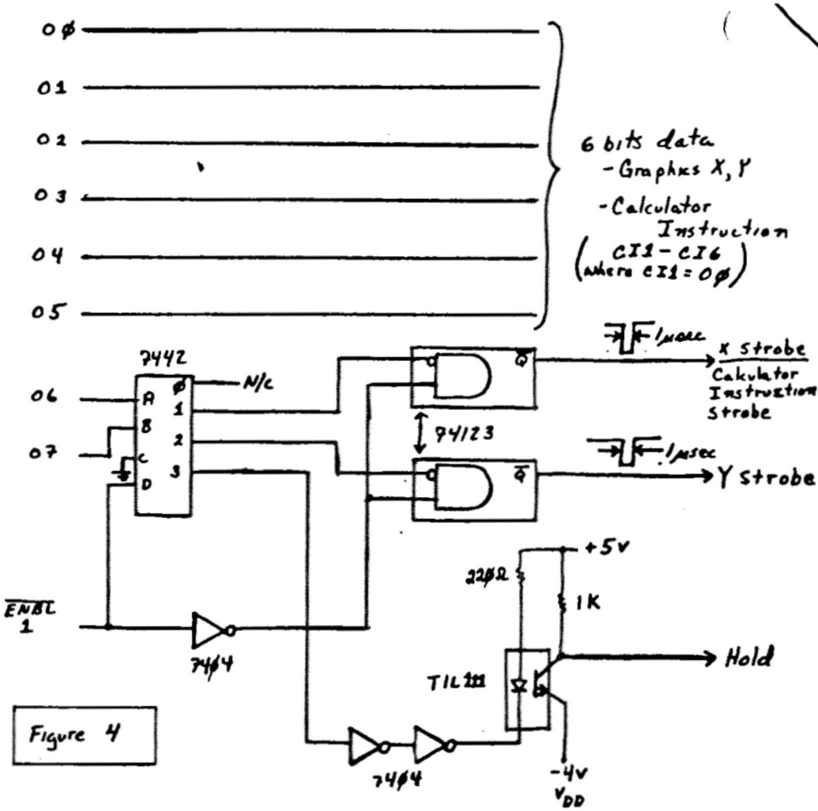
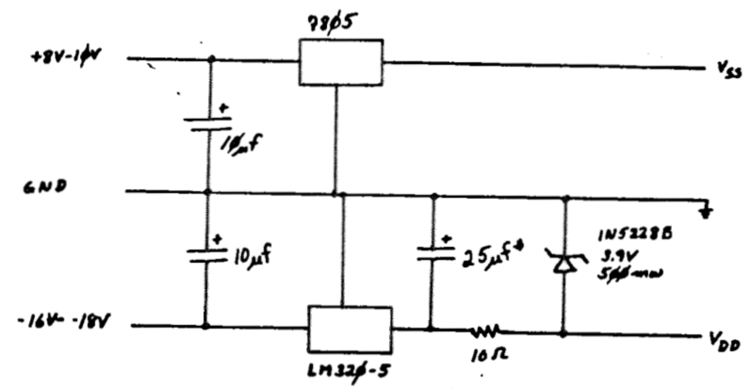


Figure 4



*Necessary for proper operation of regulator

Figure 6

My experience with this calculator chip has lead to the discovery of only one unusual feature. It appears that the flag outputs are only valid when HOLD signal is low. Other than that, everything seems to work fine.

Closing Notes

As mentioned before this information package is sufficient to get your NCU up and running. Nevertheless, it should be born in mind that this interface is flexible and the software is super simple (therefore limited). Much could be done to improve things. My current project is the development of a more substantial software package, which would turn an expanded KIM-1 into a Super programmable calculator.

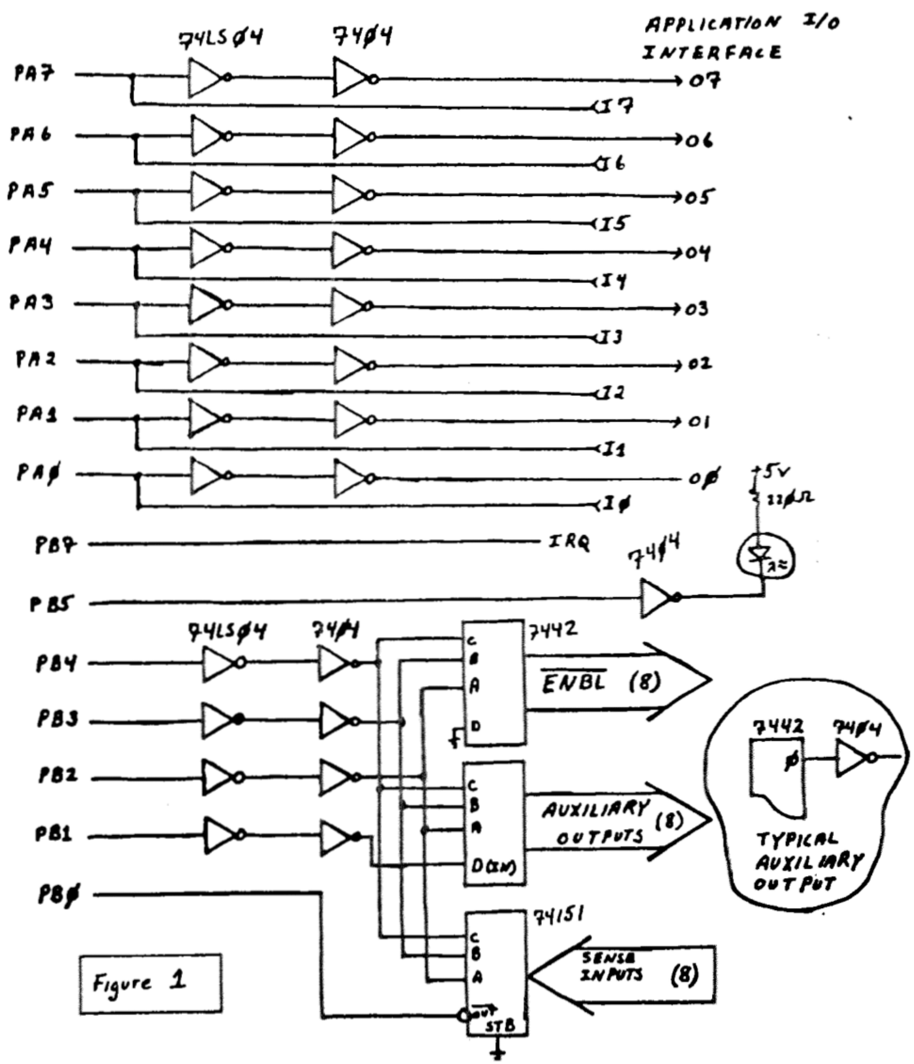


Figure 1

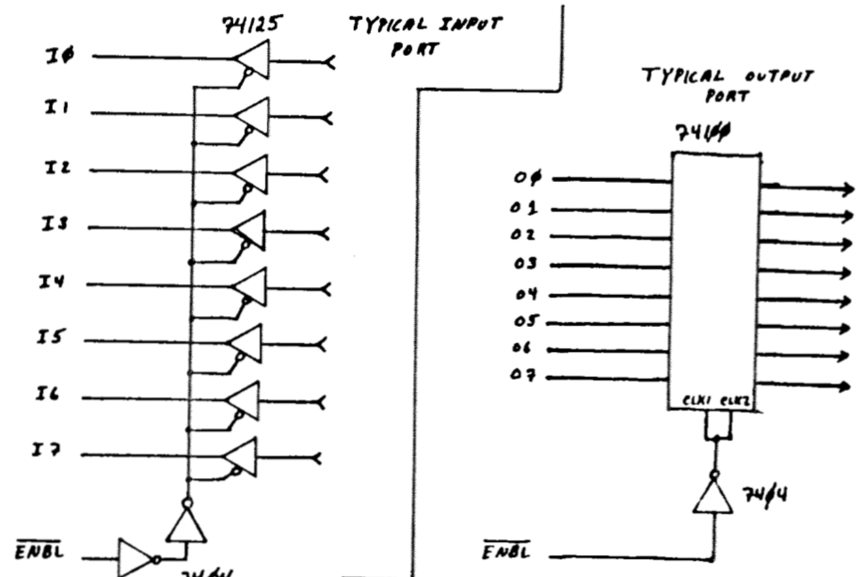


Figure 2

HEXADECIMAL OPCODES FOR NCU INSTRUCTIONS

00	0	10	TJC	20	INV	30	XEY
01	1	11	TX=0	21	EN	31	e^x
02	2	12	TX<0	22	TOGM	32	10^x
03	3	13	TXF	23	ROLL	33	x^2
04	4	14	TERR	24	SIN (SIN^{-1})	34	$\sqrt{x}$
05	5	15	JMP	25	COS (COS^{-1})	35	ln
06	6	16	OUT	26	TAN (TAN^{-1})	36	LOG
07	7	17	IN	27	SF1	37	1/x
08	8	18	SMDC	28	PF1	38	y^x
09	9	19	IBNZ	29	SF2	39	+ (M+)
0A	DP	1A	DBNZ	2A	PF2	3A	− (M−)
0B	EE	1B	XEM	2B	ECLR	3B	× (M×)
0C	CS	1C	MS	2C	RTD	3C	÷ (M÷)
0D	PI	1D	MR	2D	DTR	3D	PRW1
0E	AIN	1E	LSH	2E	POP	3E	PRW2
0F	HALT	1F	RSH	2F	MCLR	3F	NOP

'HEXADAISY' BY E&L PFEIFFER COMPUTER PRODUCTS

Perhaps the biggest pain in hand-assembly and most prone to errors is the calculation of relative branches. I've had more programs bomb out from this problem than any other. Texas Instruments has introduced a programmers calculator that nicely handles the problem, but at $50.00, the price/performance ratio is nowhere near where it should be unless you were going to use it for alot more than just branch calculations. KIM could, of course, be programmed to compute it's own relative branches but that would mean having a computer close-by at all times. And, as we all know, that just isn't possible. (Just ask Jim Butterfield).

If you're still reading, then chances are that you would be interested in hearing about 'HEXADAISY'. Picture two circular vinyl discs held together by a centered rivet and you'll have a good idea of what this hex calculator looks like. The instructions describe how to do hex arithmetic with 'HEXADAISY', but I feel that its branch calculating ability is by far more important and makes it well worth the $3.95 price tag. The price/performance ration of this device is also more realistic. 'HEXADAISY' is available for $3.95 (postpaid in USA) from:

E&L PFEIFFER COMPUTER PRODUCTS, Box 2624, Sepulveda, CA 91343
(Cal. residents add sales tax)

****

PREPROGRAMMED PROMS AND D/A CHIPS are available from Peter Bertelli, 5262 Yost Place, San Diego, CA 92109. Peter mentioned that he stocks the TVT-6 Scan PROM ($3.25) and the Motorola 3408 DAC chip ($3.50).

HAMS, TAKE NOTE----If you got turned by the MICROPROCESSOR CONTROLLED KEYBOARD in the January 1978 issue of HAM RADIO, then you'll be glad to know that a p.c. board is now available for that project. In case you didn't.....it uses a 6504 CPU, a couple of 1702A EPROMS, four 6111's, a 6530-005 and other misc. TTL and provides about all the flexibility you could ever expect in a CW keyboard. (Love those micro's!!!).

Anyway, like I was saying, the p.c. boards are now available from PYRAMID DATA SYSTEMS, DEPT A., 6 Terrace Ave., New Egypt, NJ. 08533. For $25.00 you get the board and documentation. Include an extra $1.50 if you want a reprint of the Ham Radio article.
73

****

RIVERSIDE ELECTRONIC DESIGN is still alive and well. They can be reached at 716-873-5306 in the evenings. Eugene Zumchak, one of the owners, said that they are still making the video and KIM expansion boards. I saw these boards at the CLEVELAND COMPUTERFEST and they looked well thought out and constructed...........ERIC

****

FORETHOUGHT PRODUCTS is now making a power supply available to power their "KIMSI" and similar machines. All outputs are unregulated and include +8 volts at 12 Amps, +16 volts at 1 Amp and -16 at 1 Amp. Input is either 110 VAC or 220 VAC. Price is $69.50 in kit form or $89.00 assembled. Get more info on this and their other KIM products at: FORETHOUGHT PRODUCTS, P.O. Box 8066, Coburg, Or 97401 503-485-8575

****

CONNECTICUT MICROCOMPUTER has announced immediate availability of an RS-232 ADAPTOR FOR KIM. In its present configuration, the adaptor converts current-loop to RS-232 (and vice-versa) but can easily be modified to convert TTL to RS-232 (and vice-versa). ADÂ, as it's called, comes completely assembled for $24.50 with drilled, plated-through solder pads for all connections, or, for $29.50 with barrier strips and screw terminals. Contact them at:
Pocono Rd., Brookfield, CT., 06804

****

MICRO-Z ELECTRONIC SYSTEMS has a version of MICRO-SOFT BASIC available for KIM. This 9K package sells for $100.00, is recorded on a standard KIM cassette, and comes with a 70 page manual on how to use Microsoft BASIC with KIM. Get in touch with Micro-Z at Box 2426, Rolling Hills, CA 90274, or call them at 213-377-1640.

****

THE 6502 PROGRAM EXCHANGE, 2920 Moana, Reno. NV 89509 has announced a number of new software packages for the 6502. These include an extended version of FOCAL, a 4K resident assembler, and a mini text editor.

The new FOCAL (FCL65E) offers 8 to 9 digit accuracy, 8-level priority interrupt handling, string variables and functions, and greater flexibility in its FOR, SET, and DO commands. The EXCHANGE indicates they have a FOCAL version of STAR TREK as well as other programs available.

More information, prices, and alist of other software (floating-point arithmetic package, disassemblers, games, and utility programs) may be obtained by sending $1.00 to the 6502 Program Exchange.

<u>SOFTWARE REVIEW</u> BY THE EDITOR

"XIM" BY PYRAMID DATA SYSTEMS

As soon as I hooked a terminal to KIM, it became apparent that the built-in TTY monitor was only a bare-bones approach and a more elegant program development tool was sorely needed. The functions that were most necessary included a more convenient way of entering and dumping HEX data, as well as a move routine and maybe a BREAK processor for debugging purposes. Luckily though, before I got too far into working up these routines for myself, a copy of something called "XIM" came to my attention. Basically "XIM" stands for Extended I/O Monitor and is a 1K extension of the KIM monitor. 17 commands are included in its arsenal (4 of which are user defineable) including such niceties as block move, search, and compare; hex dump and entry; a breakpoint routine; a relative branch calculator; etc.

"XIM" has been "ediot-proofed" very nicely and provides the operator feedback necessary for user-confidence. This feature has been sorely lacking in a number of software packages I have seen. SOFTWARE WRITERS TAKE NOTE.

The documentation is very complete, gives examples for each of the 17 commands, and provides a well-commented source listing of the program for ease of understanding.

"XIM" is available for $10.00 (manual and paper tape) or $12.00 (manual and KIM cassette) postpaid in USA from PYRAMID DATA SYSTEMS, Dept 'A', 6 Terrace Ave, New Egypt, NJ 08533.

****

MORRISON ELECTRONICS INC. announces availability of their <u>4K RAM board</u> designed expecially for KIM. According to the flyer, the assembled and tested board sells for $165.00 and is configured to mount directly below KIM on standoffs. Get more info from them at 3539 Lacon Rd., Hilliard, Oh 43026 (614-876-4408).

****

<u>WORD PROCESSING NEWSLETTER</u>

If you're into WP (or getting into WP) then you'll want to subscribe to a really nifty newsletter that's specializing in this fascinating portion of the computer field. Hard copy devices, computer hard and software and many other topics are covered in this monthly publication. Subscription rates are $12.95 for 12 issues (available only in the U.S. and within the Pan American Postal Union) from BOOKMAKERS, BOX 158, San Luis Rey, CA 92068. (They also publish a 2650 user group newsletter).

****

OPTIMAL TECHNOLOGY announces a <u>2708/2716 PROM PROGRAMMER for KIM</u>. Price of the EP-2A is $59.95 (assembled and tested) or $49.95 for the kit. Either way, you get the hardware, KIM software, and a circuit board connector. Write to them for more data at:
OPTIMAL TECHNOLOGY INC., Blue Wood 127, Earlysville, VA 22936

****

...And this next good number let's thank Timothy Bennett,
309 Mary St., Westerville, Ohio 43081

KIM-1 / 6502 USER NOTES INDEX BY SUBJECT VOLUME 1

APPLICATIONS FOR KIM

CASSETTE PROBLEMS/SUGGESTIONS

GENERAL KIMFORMATION

I/O SUGGESTIONS

SOFTWARE

ON VERIFYING PROGRAMS IN RAM

Ron Niessen
Ottawa

Ever had a program go wild and you're left wondering what got destroyed as a result? CHEK is a handy utility you can use to identify destroyed programs. CHEK calculates the checksum over a block of memory defined by BEG and FIN (inclusive).

I suggest that programs published in the KIM-1 USER NOTES have a ckecksum at the end so that readers can verify whether they've entered them into memory correctly.

To find the checksum for a program starting at 1780 and ending at 17A4 (e.g. CHEK), run CHEK with BEG=80,17 and FIN=A4,17. The display will show 17A6 FA, where FA is the checksum which must be entered at location 17A5.

Now to see if the program is intact, run CHEK with BEG=80,17 and FIN=A5,17. If the display shows 17A6 00, the program between 1780 and 17A4 and the checksum at 17A5 are intact.

```
                  ; CHECKSUM CALCULATOR
                  ; Put memory block start addr in E0,E1
                  ; Put memory block end addr in E2,E3
                  ; Processor must be in binary mode
                  ; 17FE,17FF must contain address=1C00
                  ; CHEK modifies the contents of E0,E1
1780 A900   CHEK LDA # $00      ; Initialize A (sum),
1782 A8          TAY            ; Y,
1783 18     CH1  CLC            ; and C to zero.
1784 71E0        ADC (BEG),Y    ; Add to sum.
1786 E6E0        INC BEG        ; Increment
1788 D002        BNE CH2        ; memory
178A E6E1        INC BEG+1      ; address.
178C A6E3   CH2  LDX FIN+1      ; Check to
178E E4E1        CPX BEG+1      ; see if
1790 D0F1        BNE CH1        ; current
1792 A6E2        LDX FIN        ; memory address
1794 E4E0        CPX BEG        ; equals the last
1796 D0EB        BNE CH1        ; memory address.
1798 18          CLC
1799 71E0        ADC (BEG),Y    ; Add in the last byte.
179B 85F5        STA $F5        ; Calculate
179D 98          TYA            ; the
179E 38          SEC            ; checksum:
179F E5F5        SBC $F5        ; 0 - sum .
17A1 8DA617      STA CH3        ; Store for display.
17A4 00          BRK            ; Exit to Monitor.
17A5 FA     SUM  .BYTE $FA      ; Checksum over CHEK.
17A6 00     CH3  .BYTE $00      ; Checksum for display.
            BEG = $E0   ;Memory block start address.
            FIN = $E2   ;Memory block end addr (L,H).
```

How 'bout some TTY graphics?.....can you expand on this?

GREETING CARD GENERATOR from Hardy Pottinger, 13 Pauline Ln.
Rolla, Missouri 65401

This is a program wirtten in 6502 assembly language for the KIM-1 microcomputer system. It is designed to accept a message from a console teletype terminated by a carriage return ($0D) and then interprets a simple list of picture descriptors to repeat the message in a desired pattern. The program as currently written has room for a 10 character message (including terminator). The pattern descriptor size is limited only by KIM's memory. The program resides in locations $200 through $26E. The message follows the program, and the pattern descriptor is entered at $279. Locations $263 and $264 are the descriptor table's low and high address bytes. The contents of these two locations may be changed if desired to allow a longer message text.

The descriptor is composed of a list of 7-bit counters of the form:

Ns,Nm,Ns,...,Nm,Ns,Nm,...,FF

where Ns is a 7-bit space count, and Nm is a 7-bit message count. A new line is signaled at any time by a count with a 1 in bit 0. Any count can be 0. A $FF marks the end of the descriptor and a return is made to the KIM monitor via a BRK instruction. The message is repeated if necessary to fill out each field of Nm bytes. Each line begins with an 18 space margin. This is arbitrary and can be changed by modifying the contents of location $212. This value must be at least 1.

Example:

```
80 80 80 80
00 05 05 05 05 85
00 05 05 05 05 85
00 05 05 05 05 85
00 05 05 05 05 85
05 05 05 85
05 05 05 85
05 05 05 85
05 05 05 85
00 05 05 05 05 85
00 05 05 05 05 85
00 05 05 05 05 85
00 05 05 05 05 85
80 80 80 FF
```

Produces a checkerboard pattern as shown on the sample runs. Note that if the message is too long to fill a field it is continued in the next field or on the next line.

```
                  ; GREETING CARD GENERATOR
                  ; DRAW A FIGURE COMPOSED OF TEXT FROM A USER GENERATED
                  ; MESSAGE
                  ; POINTER STORAGE
                  ;
                  CMPTR   EQU    0
                  LPTR    EQU    1
                  COUNT   EQU    2          ; 'COUNT' FROM LIST
                  MCNT    EQU    3          ; 7-BIT COUNT FROM 'COUNT'
                  GETCH   EQU    $1E5A      ; GET CHAR ROUTINE
                  OUTSP   EQU    $1E9E      ; OUTPUT SPACE ROUTINE
                  OUTCH   EQU    $1EA0      ; OUTPUT CHAR ROUTINE
                  ;
                          .LOC   $200
                  ;
0200 A2 00        START   LDX    #$00       ; CLEAR X REG
0202 86 00                STX    CMPTR      ; RESET POINTERS
0204 86 01                STX    LPTR
                  ;
0206 20 5A 1E     GMSG    JSR    GETCH      ; GET CHAR FROM TTY
0209 9D 6F 02             STA    MSG,X      ; STORE IN MESSAGE AREA
020C E8                   INX               ; INCR X REG
020D C9 0D                CMP    #$0D       ; = CR?
020F D0 F5                BNE    GMSG       ; GET MORE IF NOT CR
                  ;
                  ; OUTPUT OFFSET # OF SPACES FOR LEFT MARGIN
                  ;
0211 A2 12        LMARG   LDX    #18
0213 20 9E 1E     LMRG1   JSR    OUTSP      ; DO LEFT MARGIN
0216 CA                   DEX
0217 D0 FA                BNE    LMRG1
                  ;
                  ; GET COUNT OF SPACE FIELD
                  ;
0219 20 5E 02     SLIST   JSR    GCNT
021C AA                   TAX               ; COUNT TO X REG
021D F0 06                BEQ    SP2        ; GO TO SP2 IF COUNT=0
021F 20 9E 1E     SP1     JSR    OUTSP
0222 CA                   DEX
0223 D0 FA                BNE    SP1
0225 A9 80        SP2     LDA    #$80
0227 24 02                BIT    COUNT      ; TEST COUNT FOR END FLAG
0229 D0 26                BNE    ENDSC      ; END OF DESCRIPTOR
                  ;
                  ; GET COUNT AND DO MESSAGE FIELD
                  ;
022B 20 5E 02             JSR    GCNT
022E F0 1B                BEQ    EMSG
0230 85 03                STA    MCNT       ; SAVE COUNT IN MCNT
0232 A6 00        MS1     LDX    CMPTR      ; GET CURRENT MESSAGE POINTER
0234 BD 6F 02     MS2     LDA    MSG,X      ; GET CURRENT MESSAGE BYTE
0237 C9 0D                CMP    #$0D       ; TEST FOR CR
0239 D0 06                BNE    MS3        ; NOT A CR
023B A2 00                LDX    #$00       ; CLEAR X
023D 86 00                STX    CMPTR      ; RESET MESSAGE POINTER
023F F0 F3                BEQ    MS2
0241 E8           MS3     INX
0242 20 A0 1E             JSR    OUTCH
0245 E6 00                INC    CMPTR
0247 C6 03                DEC    MCNT       ; TEST COUNT
0249 D0 E7                BNE    MS1        ; DO MORE IF NOT ZERO
                  ;
                  ; END OF MESSAGE FIELD
                  ;
024B A9 80        EMSG    LDA    #$80
024D 24 02                BIT    COUNT      ; TEST FOR END OF LINE
024F F0 C8                BEQ    SLIST      ; DO NEXT SPACE FIELD
                  ;
                  ; END OF DESCRIPTOR (LINE)
                  ;
0251 A9 0D        ENDSC   LDA    #$0D       ; DO CR/LF
0253 20 A0 1E             JSR    OUTCH
0256 A9 0A                LDA    #$0A
0258 20 A0 1E             JSR    OUTCH
025B 4C 11 02             JMP    LMARG
                  ;
                  ; GCNT
                  ; GET COUNT FROM LIST
                  ; ORIGINAL COUNT IN 'COUNT' AND A
                  ; 7-BIT COUNT IN ACC
                  ;
025E A6 01        GCNT    LDX    LPTR       ; GET CURRENT LIST POINTER
0260 E6 01                INC    LPTR       ; BUMP IT
0262 BD 79 02             LDA    LBASE,X    ; GET CURRENT LIST ELEMENT
0265 C9 FF                CMP    #$FF       ; TEST FOR END OF LIST
0267 D0 01                BNE    GCNT1      ; NOT END
0269 00                   BRK               ; END OF LIST
026A 85 02        GCNT1   STA    COUNT      ; SAVE ORGINAL COUNT
```

```
026C
026C 29 7F   297F      AND     #$7F      ; MASK OFF BIT0
026E 60                RTS               ; RETURN

             MSG       RMB     10        ; MESSAGE AREA
             LBASE     RMB     1         ; CARD DESIGN DESCRIPTION GOES H

                       END
0  UNDEFINED SYMBOLS **
```

HERE'S AN EXAMPLE.........

```
KIM        *a
0000 06 200
0200 A2 G12345
```

```
12345     12345     12345
12345     12345     12345
12345     12345     12345
12345     12345     12345
     12345     12345
     12345     12345
     12345     12345
     12345     12345
12345     12345     12345
12345     12345     12345
12345     12345     12345
12345     12345     12345
```

end

Here's some interesting comments and a neat idea from Ian Thurston 22 Concord Ave., Dundas, ONT, Canada (L9H 1R6)....

Right now, I'm using my KIM to train music students to recognise different musical intervals, and the results are fantastic ! One student who didn't know a diminished fifth from an empty beer bottle a few weeks ago has made really good progress, largely because he enjoys using the KIM trainer program.

I'm working on a game now which looks promising. The premise is that you, the player, are in a submarine represented on the display by a vertical line. You can control your depth, which is fortunate, because every now and then, a subchaser quickly appears on the left side of the display (the surface), drops a depth charge, and scoots. If you happen to be at the position where the depth charge explodes (unpredictably, of course !), tough cheese ! Otherwise, the game continues. On the other hand, if you launch a torpedo quickly enough, you may sink the subchaser, and win .

In the meantime, though, I thought you might be interested in the enclosed note on how I use a voice-operated-relay with my KIM as an input device.

<u>VOCAL INPUT TO KIM</u>

Try using a simple Voice-Operated-Relay (VOR) circuit as an input device. With a little ingenuity, you can use a VOR not only as a go/no-go input, but also as a variable input.

I hooked a VOR kit I had lying around (Radio Shack 28-131) to application pin 8 (PA 7). Now, using the BMI and BPL instructions, I'm able to poll the relay. For example, the following routine polls the relay for about 2 seconds. If there is no voice command, it exits with A = ØØ; if a voice command closes the relay, the routine exits within ¼second with A = Ø1.

```
        LDX #$Ø9    A2 Ø9       Load counter for 8 ¼ sec. intervals .
TIMES   LDA #$ØØ    A9 ØØ       Set A = ØØ in case of no response
        DEX         CA          ¼ sec. counted .
        BMI EXIT    3Ø 11       If all done, leave with A = ØØ ;
        LDA #$FF    A9 FF       if not, load timer
        STA 17Ø7    8D Ø7 17    to time about ¼ sec.
TIME    LDA 17Ø7    AD Ø7 17    Poll timer
        BPL TIME    1Ø FB       until done,
        LDA PAD     AD ØØ 17    then check Data Port A .
        BMI TIMES   3Ø EC       Keep timing unless relay closed,
        LDA #$Ø1    A9 Ø1       in which case, set A = Ø1
EXIT    (RTS        6Ø, or continue)
```

This program assumes a relay connected in the normally-open mode, with a closure of at least ¼ second.

Not bad, for under $10, but there are more possibilities. Here's one application that allows <u>variable</u> inputs using a VOR. How ? Simply by timing how long the VOR remains closed.

```
        LDX #$ØØ     A2 ØØ       Initialise counter.
        STX PADD     8E Ø1 17    Set Data Direction to Input
SETYET  LDA PAD      AD ØØ 17    Check Data Reg. A
        BMI SETYET   3Ø FB       until voice begins ;
DELAY   LDA #$FF     A9 FF       then load timer
        STA 17Ø7     8D Ø7 17    to time about ¼ sec.
TIMER   LDA 17Ø7     AD Ø7 17    Check timer
        BPL TIMER    1Ø FB       until done,
        INX          E8          then increment ¼sec. counter.
        LDA PAD      AD ØØ 17    Check that relay is still on .
        BPL DELAY    1Ø FØ       If so, go time some more.
        ?   ?        XX XX       If not, leave with count in X reg.
```

With a little experimentation, you'll find it's possible to control the length of time the relay stays closed by controlling what you say. With my set-up, I've found that quickly saying "one" produces a count of Ø1 in the X register. Saying " one-two " produces a count of Ø2, and so on. Of course, the system isn't elegant, nor is it 100 % reliable. But it sure is fun ! (And incidentally, a good way of answering those smart alecks who ask you if your computer can talk yet !)

NOTE: To make the above routine work with my VOR, I had to disable an RC network that latched the relay "on" for a few seconds.

end

Do you remember what day you were born on? Here's an interesting diversion from ...Harvey Heinz, 9730 Townline Diversion, Surrey, B.C. V3V 2T2 Canada

This program will compute the day of the week for any date between Sept.14, 1752 (the start of the Gregarian calender in the British colonies) and Dec.31, 1999.

Enter 2 digits for month in loc.0001.-- 2 digits for date in 0002.
Century in location 0003, and 2 digits for year in 0004.

Press + and GO. Answer will appear in location 0000 as a 2 digit number .
01=Sunday, 02=Monday,03=Tuesday, etc.,to 00=Saturday.

```
0000   xx ?? ?? ?? ?? F8 38 A9 00 85 00 85 A0 85 B3 AA
0010   A5 04 C9 00 F0 18 C9 04 90 14 A8 8A 18 69 01 AA
0020   98 38 E9 04 D0 F0 A5 01 C9 03 B0 02 C6 B3 A5 03
0030   C9 20 B0 67 C9 19 F0 2C C9 18 90 06 A9 02 85 00
0040   D0 22 C9 17 90 55 A5 04 C9 53 90 06 A9 04 85 00
0050   D0 12 C9 52 D0 45 A5 01 C9 09 90 3F A5 02 C9 14
0060   90 39 B0 E8 8A 18 65 00 65 02 A6 01 75 A0 24 B3
0070   F0 03 38 E9 01 18 65 04 90 02 C6 A0 38 B0 02 E9
0080   07 C9 07 B0 FA 24 A0 10 07 E6 A0 18 69 01 D0 EF
0090   85 00 A9 00 85 FA 85 FB 4C 4F 1C A9 88 D0 F1 xx
00A0   xx 01 04 04 00 02 05 00 03 06 xx xx xx xx xx xx
00B0   01 04 06 xx
00B4   last location + 1
```

EXAMPLE:
Dec 7, 1941

$0001 = 12
$0002 = 07
$0003 = 19
$0004 = 41
+ GO

If you attempt to enter a date outside of the limits, the program will put 88 in location 0000.

The program uses this equation:

$$W \equiv M + D + C + 1\tfrac{1}{4}Y \quad (\mathrm{Mod.}\ 7)$$

W= day of the week (01 = Sun.,02 = Mon., etc., 00 = Sat.)
M= special number for month
D= Date (day of the month)
C= special number for century
Y= year (of century)

end

HERE'S A KLUGE-HARP LOADER FOR YOU MUSIC FREAKS, FROM:

R. S. McEvoy
46 Browallia Crescent
Loftus 2232 N.S.W.
Australia

"Ron Kushnier's Harp in #6 is a real improvement but lacks the ability to take rests-silence is important in real music. I'm sending you a simple patch which treats code #FF as a rest.

Also included is a Kluge Harp Loader which uses a TVT as an input terminal. Not elegant but it does allow direct loading from sheet music to memory W/O all the table look-up.

Possibly the most important feature is the note codes - they're right, by tuning fork & frequency meter. Now you can play duets with KIM.

Upcoming projects include a music transcriber to automatically take care of sharps & flats in going from one key to another. Also, a hardware multiplexed bus system to allow KIM to play chords. How about some articles on music or sound in general".

```
Ø3ØØ A2 ØØ               LDX #ØØ       INDEX TO SCORE START
  Ø2 AØ ØØ               LDY #ØØ       SET FOR LOW OCTAVE
  Ø4 8C EB Ø3  NSTART    STY TEMPY
  Ø7 2Ø 5A 1E            JSR GETCH     GET KB INPUT
  ØA C9 Ø8               CMP '←'       IF IT IS '←' KEY, INDEX
  ØC DØ Ø8               BNE 1         BACK ONE COUNT, DISPLAY
  ØE CA                  DEX           NEW INDEX AND
  ØF 8A                  TXA           RETURN
  1Ø 2Ø 3B 1E            JSR PRTBYT
  13 2Ø EC Ø3            JSR LFCR
  16 4C Ø2 Ø3            JMP NSTART        -OR-
  19 C9 1F      1        CMP '→'       IF IT IS '→' KEY, INDEX
  1B DØ ØB               BNE 2         FORWARD ONE COUNT,
  1D E8                  INX           DISPLAY NEW INDEX
  1E 8A                  TXA           AND RETURN
  1F 2Ø 3B 1E            JSR PRTBYT
  22 2Ø EC Ø3            JSR LFCR
  25 4C Ø2 Ø3            JMP NSTART        -OR-
  28 C9 7Ø      2        CMP 'P'       IF IT IS 'P' KEY, NEXT
  2A DØ Ø7               BNE 3         2 KEY INPUTS ARE LOADED
  2C 2Ø 9D 1F            JSR GETBYT    DIRECTLY TO INDEXED LOC.
  2F DØ 57               BNE 6
  31 FØ 55               BEQ 6             -OR-
  33 C9 68      3        CMP 'H'       IF IT IS 'H' KEY, NEXT
  35 DØ Ø8               BNE 4         LOCATION WILL LOAD FROM
  37 AØ ØD               LDY #ØD       HIGH OCTAVE
  39 8C EB Ø3            STY TEMPY
  3C 2Ø 5A 1E            JSR GETCH         -OR-
  3F AC EB Ø3   4        LDY TEMPY
  42 C9 61               CMP 'A'       COMPARE TO 'A' KEY. IF A MATCH
  44 FØ 3F               BEQ 5         LOAD 'A' CODE. OTHERWISE,
  46 C8                  INY           INC. INDEX FOR NEXT NOTE.
  47 C9 41               CMP 'A#'        ETC. FOR ALL POSSIBLE
  49 FØ 3A               BEQ 5           NOTES.
  4B C8                  INY
  4C C9 62               CMP 'B'
  4E FØ 35               BEQ 5
  5Ø C8                  INY
  51 C9 63               CMP 'C'
  53 FØ 3Ø               BEQ 5
  55 C8                  INY
  56 C9 43               CMP 'C#'
  58 FØ 2B               BEQ 5
  5A C8                  INY
Ø35B C9 64               CMP 'D'
  5D FØ 26               BEQ 5
  5F C8                  INY
  6Ø C9 44               CMP 'D#'
  62 FØ 21               BEQ 5
  64 C8                  INY
  65 C9 65               CMP 'E'
  67 FØ 1C               BEQ 5
  69 C8                  INY
  6A C9 66               CMP 'F'
  6C FØ 17               BEQ 5
  6E C8                  INY
  6F C9 46               CMP 'F#'
  71 FØ 12               BEQ 5
  73 C8                  INY
  74 C9 67               CMP 'G'
  76 FØ ØD               BEQ 5
  78 C8                  INY
  79 C9 47               CMP 'G#'
  7B FØ Ø8               BEQ 5
  7D C8                  INY
  7E C9 72               CMP 'R'       COMPARE TO 'REST' KEY, IF
  8Ø FØ Ø3               BEQ 5         MATCH, WILL LOAD #FF
  82 2Ø Ø2 Ø3            JMP NSTART    NOT VALID KEY, KEEP TRYING
  85 B9 5Ø Ø2   5        LDA NOTE,Y    GET NOTE VALUE FROM TABLE
  88 9D ØØ ØØ   6        STA TUNE,X    AND STORE IN SCORE.
  8B A9 2Ø               LDA 'SP'      PUT A SPACE ON CRT
  8D 2Ø AØ 1E            JSR OUTCH
  9Ø 8A                  TXA           THEN OUTPUT PRESENT
  91 2Ø 3B 1E            JSR PRTBYT    LOCATION.
  94 2Ø EC Ø3            JSR LFCR      DO CRLF W/O SCREWING UP X
  97 E8                  INX           ADVANCE TO NEXT SCORE LOC.
  98 2Ø 5A 1E   8        JSR GETCH     GET KB INPUT
  9B AØ ØØ               LDY #ØØ       SET TIME INDEX
  9D C9 31               CMP '1'       COMPARE TO "WHOLE NOTE" KEY.
  9F FØ 34               BEQ 7         IF A MATCH, LOAD '1' CODE.
  A1 C8                  INY           OTHERWISE, INC INDEX FOR
  A2 C9 21               CMP '1.'      NEXT TIME SIGNATURE.
  A4 FØ 2E               BEQ 7           ETC FOR ALL LISTED
  A6 C8                  INY             TIME SIGNATURES.
  A7 C9 32               CMP '2'
  A9 FØ 2A               BEQ 7
  AB C8                  INY
  AC C9 4Ø               CMP '2.'
  AE FØ 25               BEQ 7
  BØ C8                  INY
Ø3B1 C9 34               CMP '4'
  B3 FØ 2Ø               BEQ 7
  B5 C8                  INY
  B6 C9 24               CMP '4.'
  B8 FØ 1B               BEQ 7
  BA C8                  INY
  BB C9 38               CMP '8'
  BD FØ 16               BEQ 7
  BF C8                  INY
  CØ C9 2A               CMP '8.'
```

```
C2  FØ 11                 BEQ ⑦
C4  C8                    INY
C5  C9      C936          CMP '6'        (1/16)
C7  FØ  .C  FØØC          BEQ ⑦
C9  C8                    INY
CA  C9 5E                 CMP '6.'       (1/16.)
CC  FØ Ø7                 BEQ ⑦
CE  C8                    INY
CF  C9 33                 CMP '3'        (TRIPLET)
D1  FØ Ø2                 BEQ ⑦
D3  DØ C3                 BNE ⑧
D5  B9 7Ø Ø2       ⑦      LDA TIME,Y     GET TIME VALUE FROM TABLE
D8  A9 2Ø                 STA TONE, X    AND STORE IN SCORE
DB  A9 2Ø                 LDA 'SP'       PUT A SPACE ON SCREEN.
DD  2Ø AØ 1E              JSR OUTCH
EØ  8A                    TXA            THEN OUTPUT PRESENT
E1  2Ø 3B 1E              JSR PRTBYT     STORAGE LOCATION.
E4  2Ø EC Ø3              JSR LFCR
E7  E8                    INX            ADVANCE TO NEXT LOC.
E8  2Ø Ø2 Ø3              JMP NSTART     THEN START AGAIN.
EB  XX                    TEMPY
EC  A9 ØA          LFCR   LDA LF         SUB TO OUTPUT LF & CR
EE  2Ø AØ 1E              JSR OUTCH      W/O EFFECTING X.
F1  A9 ØD                 LDA CR
F3  2Ø AØ 1E              JSR OUTCH
F6  6Ø                    RTS
```

...AND NOW, THE NOTE TABLE---

Ø25Ø DF	NOTE	A		Ø27Ø 2Ø	TIME	1/1
51 D3		A#		71 3Ø		1/1.
52 C6		B		72 1Ø		1/2
53 BA		C		73 18		1/2.
54 B1		C#		74 Ø8		1/4
55 A6		D		75 ØC		1/4.
56 9C		D#		76 Ø4		1/8
57 93		E		77 Ø6		1/8.
58 8A		F		78 Ø2		1/16
59 83		F#		79 Ø3		1/16.
5A 7B		G		7A XX		TRIPLET
5B 74		G#				
5C FF		REST				
5D 6E		A	HIGH OCTAVE			
5E 67		A#				
5F 61		B				
6Ø 5C		C				
61 56		C#				
62 51		D				
63 4C		D#				
64 48		E				
65 44		F				
66 4Ø		F#				
67 3C		G				
68 38		G#				
69 FF		REST				

PATCH TO RONALD KUSHNIER'S KLUGE HARP TO INCORPORATE 'RESTS'

```
Ø231  A9 ØØ         LDA ØØ       RESET POSSIBLE PREVIOUS REST
      8D 12 Ø2      STA Ø212
      C8            INY
      C8            INY
      B9 ØØØØ       LDA ØØ,Y     TEST NOTE FOR END OF SCORE
      C9 ØØ         CMP #ØØ
      FØ C6         BEQ Ø2Ø2     YES: PLAY IT AGAIN, KIM
      C9 FF         CMP # FF     IS IT A REST?
      DØ C4         BNE ⑤        NO: CONTINUE PLAYING
      A9 Ø2         LDA #Ø2      YES: SILENCE PA OUTPUT
      8D 12 Ø2      STA Ø212
      DØ BD         BNE ⑤        UNCONDITIONAL JMP(CONTINUE)
```

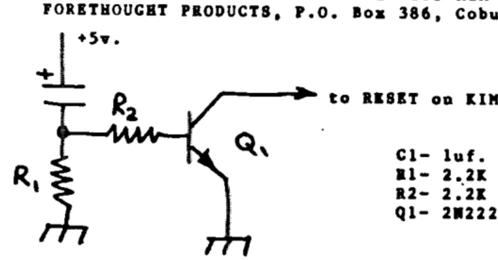
NOTES ON USING KLUGE HARP LOADER

1. LOAD NOTES USING KEYS A - G.
2. LOAD TIME VALUES W/ FOLLOWING KEYS:

WHOLE	- 1	EIGHTH	- 8		
HALF	2	SIXTEENTH	- 6	TRIPLET	- 3
QUARTER	4	REST	- R		

3. TO SHARPEN A NOTE, SHIFT IT.
4. TO EXTEND A TIME VALUE BY 1/2 (DOT IT), SHIFT IT.
5. STEP FORWARD W/ → KEY, BACKSTEP W/ ← (IF YOUR KB LACKS THESE KEYS, ANY KEYS WILL DO)
6. FOR HIGH OCTAVE, HIT THE 'H' KEY BEFORE NOTE KEY.
7. TO ENTER ODD VALUES, IE: A NOTE OUTSIDE 2 OCTAVES, A HALF NOTE TIED TO A DOTTED HALF ETC. USE THE 'P' KEY. THE FOLLOWING TWO KEY ENTRIES LOAD AS A BYTE INTO OPEN LOCATION.

Here's a POWER ON RESET CIRCUIT for KIM from Leonard Crane, FORETHOUGHT PRODUCTS, P.O. Box 386, Coburg, Ore, 97401



C1- 1uf. @15v.
R1- 2.2K ohm
R2- 2.2K ohm
Q1- 2N2222 (or equiv.)

KIM Program: DICEY Jan/77 Jim Butterfield, Toronto

This program rolls dice. Quietly. If you have an urge to play a dice game like Yahtze at 3 a.m. you won't wake the household. You can specify how many dice in COUNT, address 029E; from one to six - five are used in the program listing.

To roll all dice, hit GO. To roll selected dice only, hit keys 1 to 6 to indicate which ones you want, then hit GO. Many games need this kind of selective roll: Yahtze, Poker Dice, Ship/Captain/Crew.

Ship/Captain/Crew, for example, allows you three rolls per play, using five dice. A six is your ship; if you don't have one, you must roll all dice again. Once you have a ship, look for a five, which is your captain; if you don't have him, roll everything except the ship. When you have both ship and captain the total of the remaining dice is your crew, which is your score. You may try to improve your crew if you have any rolls left.

```
0200 D8          START  CLD
0201 20 40 1F           JSR KEYIN     directional register
0204 20 6A 1F           JSR GETKEY    test key input
0207 AE 9E 02           LDX COUNT     how many dice?
020A CA                 DEX           minus one for loop counter
020B 86 90              STX CNT
020D C9 13              CMP #$13      GO key?
020F D0 30              BNE NOGO      no, skip Roll procedure
0211 B5 A0       VUE    LDA FLAG,X    yes, test ...
0213 D0 0A              BNE RUN       any dice rolling?
0215 CA                 DEX
0216 10 F9              BPL VUE
0218 A6 90              LDX CNT
021A F6 A0       VEX    INC FLAG,X    no; roll 'em all
021C CA                 DEX
021D 10 FB              BPL VEX
021F A4 90       RUN    LDY CNT       random values for each die
0221 38          ROLL   SEC           ..whether used or not
0222 A5 97              LDA RND+1
0224 65 9A              ADC RND+4
0226 65 9B              ADC RND+5
0228 85 96              STA RND       new random value
022A A2 04              LDX #4
022C B5 96       RLP    LDA RND,X
022E 95 97              STA RND+1,X
0230 CA                 DEX
0231 10 F9              BPL RLP
0233 29 07              AND #$07      change these lines...
0235 C9 06              CMP #6         .. for n-sided dice
0237 B0 E8              BCS ROLL      reject this number?
0239 99 A6 00           STA NUMB,Y    store new roll
023C 88                 DEY
023D 10 E2              BPL ROLL
023F 30 45              BMI PLACE
0241 AA CA       NOGO   TAX DEX       test input key
0243 EC 9E 02           CPX COUNT     legal?
0246 B0 04              BCS NOKEY     no, ignore
0248 A9 01              LDA #1        set "roll" flag
024A 95 A0              STA FLAG,X
024C A9 7F       NOKEY  LDA #$7F      open display
024E 8D 41 17           STA SADD
0251 A2 05              LDX #5        six digits
0253 A9 00              LDA #0        blank unwanted dice
0255 A0 13              LDY #$13      right-hand digit
0257 EC 9E 02    LITE   CPX COUNT     stay blank?
025A B0 08              BCS DARK      yes, skip next part
025C B5 A0              LDA FLAG,X
025E F0 02              BEQ FLITE
0260 F6 AC              INC WINDOW,X  roll display
0262 B5 AC       FLITE  LDA WINDOW,X
0264 8D 40 17    DARK   STA SAD
0267 8C 42 17           STY SBD
026A C6 91       STALL  DEC ZIP
026C D0 FC              BNE STALL
026E 88 88 CA           DEY DEY DEX
0271 10 E4              BPL LITE
0273 A5 92              LDA TIMER     are we rolling?
0275 F0 89              BEQ START     no, test keys
0277 C6 92              DEC TIMER     time out the roll
0279 D0 D1              BNE NOKEY     not time yet?
027B A6 93              LDX DIE       which die stops?
027D B4 A6              LDY NUMB,X    what number is rolled?
027F B9 E8 1F           LDA TABLE+1,Y change to segments
0282 95 AC              STA WINDOW,X  and put into display window
0284 D6 A0       WIPE   DEC FLAG,X    clear flag
0286 D0 FC              BNE WIPE       ..for sure
0288 A2 00       PLACE  LDX #0
028A B5 A0       PLAY   LDA FLAG,X    search for..
028C D0 08              BNE NEXT      ..next rolling die
028E E8                 INX
028F EC 9E 02           CPX COUNT
0292 D0 F6              BNE PLAY
0294 F0 B6              BEQ NOKEY     none rolling; quit
0296 A9 50       NEXT   LDA #$50      timing
0298 85 92              STA TIMER
029A 86 93              STX DIE       record next roller
029C D0 AE              BNE NOKEY     and keep going
029E 05          COUNT  .BYTE 5       5 dice
```

TEASER (Shooting Stars) - Jumbo version Jim Butterfield, Toronto

Same rules as for Bob Albrecht's original Teaser; but with a random starting pattern. The object is to invert the starting pattern; so if the board starts out with all nine positions lit, your mission is to turn them all off. If you happen to start with only one position lit, you must try to light all the others.

```
1 2 3
4 5 6
7 8 9
```

When you accomplish this, the display will signal that you've won. Pressing GO will then give you a new, random, game. If you press GO before you've won, it will take you back to the start of the game you were doing.

Identity of the various positions is shown in the chart at upper right. The usual rules apply: you can select only lit positions, and they will invert all segments in their field of influence. For example, position 5 inverts 2, 4, 5, 6, and 8; position 2 inverts only 1, 2, and 3.

If you want to play a particular board, you can set it up in "segment" form in locations BORD to BORD+2 (addresses 0080 to 0082) and then start the program at BEGIN, location 0217.

```
0200 E6 83       START  INC SEED      scramble random number
0202 20 40 1F           JSR KEYIN      ..while GO key is down
0205 D0 F9              BNE START
0207 A2 02              LDX #2        for each digit position..
0209 A5 83              LDA SEED       set random..
020B 48          IP1    PHA
020C 29 49              AND #$49        ..horizontal segments
020E 95 80              STA BORD,X       ..into board
0210 68                 PLA          recall random number
0211 4A                 LSR A         and shift
0212 09 80              ORA #$80       setting bit 7
0214 CA                 DEX
0215 10 F4              BPL IP1
```

more TEASER

```
          A9 06     ; enter here if BORD is pre-set
0217 A9 06          BEGIN  LDA #6
0219 85 84   85 84         STA WINDOW       create  a frame
021B A9 30   A9 30         LDA #$30           for the board
021D 85 88                 STA WINDOW+4
021F A5 84          GO     LDA WINDOW       has this game been won?
0221 C9 06                 CMP #6
0223 D0 DD                 BNE START        yes, make new board
0225 A2 02                 LDX #2           no, copy board into window
0227 B5 80          LP2    LDA BORD,X
0229 95 85                 STA WINDOW+1,X
022B CA                    DEX
022C 10 F9                 BPL LP2
022E A0 11          TOP    LDY #$11         initial digit pointer
0230 A2 04                 LDX #4           five digits
0232 A9 7F                 LDA #$7F         directional register
0234 8D 41 17              STA PADD
0237 B5 84          LITE   LDA WINDOW,X
0239 8C 42 17              STY SBD
023C 8D 40 17              STA SAD
023F A9 7F                 LDA #$7F        delay
0241 E9 01          ZIP    SBC #1
0243 D0 FC                 BNE ZIP
0245 8D 42 17              STA SBD         store zero to clear display
0248 88 88                 DEY DEY         set up next ..
024A CA                    DEX               .. display position
024B 10 EA                 BPL LITE
024D 20 40 1F              JSR KEYIN       set directnl reg to input
0250 D8                    CLD
0251 20 6A 1F              JSR GETKEY       key depressed?
0254 C9 13                 CMP #$13         GO key?
0256 F0 C7                 BEQ GO           yes, do GO procedure
0258 C9 0A                 CMP #$0A         no key or greater than 9?
025A B0 D2                 BCS TOP          yes, return to display
025C AA CA                 TAX DEX          set X=key - 1
025E 30 CE                 BMI TOP          zero key?  skip.
0260 86 89                 STX TEMP         = value 0 to 8
0262 A0 03                 LDY #3
0264 88             KEY    DEY              divide X by 3 to give:
0265 CA CA CA              DEX DEX DEX
0268 10 FA                 BPL KEY
026A B9 9E 02              LDA MASK,Y       ..segment ID in Y
026D 35 88                 AND WINDOW+4,X    ..digit ID in X (negtv)
026F F0 BD                 BEQ TOP            illegal move - return
0271 A5 89                 LDA TEMP        Ready to make move:
0273 0A                    ASL A             Multiply (key-1) by 3
0274 65 89                 ADC TEMP           to set Move Table pointer
0276 A8                    TAY                 into register Y
0277 A9 49                 LDA #$49          Set up flag for win test
0279 85 89                 STA TEMP
027B A2 02                 LDX #2          Make move by..
027D B5 85          CRN    LDA WINDOW+1,X
027F 59 A1 02              EOR TABL,Y        ..EOR'ing move table
0282 95 85                 STA WINDOW+1,X      ..into display
0284 55 80                 EOR BORD,X
0286 25 89                 AND TEMP           Update win-test flag
0288 85 89                 STA TEMP
028A C8 CA                 INY DEX            on to the next digit
028C 10 EF                 BPL CRN
028E A5 89                 LDA TEMP         Now test for win
0290 C9 49                 CMP # $49         all segments OK?
0292 D0 9A                 BNE TOP             nope, return
0294 05 84                 ORA WINDOW       Add win signal to display
0296 85 84                 STA WINDOW
0298 A9 79                 LDA #$79
029A 85 88                 STA WINDOW+4
029C D0 90                 BNE TOP
029E 08 40 01 MASK         .BYTE 8,40,1
02A1 00 41 41 01 01 01 41 41 00  TABL  .BYTE 0,41,41,1,1,1,41,41,0,0,0,49
02AA 00 00 49 40 49 40 49 00 00               40,49,40,49,0,0,0,48,48,
02B3 00 48 48 08 08 08 48 48 00               8,8,8,48,48,0
02BC end
```

Notes on Jumbo TEASER (Shooting Stars)

Bored by regular TEASER, now that you've figured out the moves? Jumbo TEASER gives you a new problem every time. And each problem is tough - maybe you've forgotten how hard the original game was until you memorized the solution.

Every position generated by the program is solvable, although some are devilishly hard to get. Make a note of the original board diagram - it's easy to forget - together with the desired winning pattern, like this:

```
Original board:  * . *        Win on:  . * .
                 * * .                 . . *
                 * . *                 . * .
```

The example above can be solved in five moves .. but you can poke around for hundreds of moves trying to find that combination!

To set up the original game of teaser, if you want it, the following coding will do:

```
(anywhere in memory)  A9 40     LDA #$40
                      85 81     STA BORD+1
                      A9 00     LDA #0
                      85 80     STA BORD
                      85 82     STA BORD+2
                      4C 17 02  JMP BEGIN
```

If you locate the above coding at 0200 to 020C, the program will play only the "standard" game. Locate it elsewhere, and the first game will be standard; after that, anything goes!

For those who have forgotten the moves, here are the areas of influence for each key:

```
1 * .   * 2 *   . * 3   * . .   . * .   . . *   . . .   . . .   . . .
* * .   . . .   . * *   4 . .   * 5 *   . . 6   * * .   . . .   . * *
. . .   . . .   . . .   * . .   . * .   . . *   7 * .   * 8 *   . * 9
```

Here are some interesting comments from John Crossley...

"... I've been going to the Sacramento Microcomputer Users Group meetings for several months but last month I found at least four 6502 people. I told them about you and one told me that he has already sent in a subscription. It's nice not to be alone.

I sent away to the 6502 Program Exchange and got FOCAL-65 and a really nice disassembler. The disassembler is one of the slickest pieces of software that I've seen, well worth the 35¢. FOCAL-65 is an interesting language to use. The only problem is that the execution speed is slow. The June Kilobaud published a comparison of the speed of various BASICS and FOCAL was six times slower than the slowest. The nice features are the one dimensional arrays and the fact that the commands can be abreviated to one letter.

I've got my KIMSI!! It came in the mail one day and was running the next. The reason that it wasn't running that night is the not-soldered joint. My only reservation is the way that they handle the I/O

ports. First they use F000-FFFA which means that I can't use the KIMATH without relocating the whole program. Secondly, since some S-100 I/O boards use the upper 8 bits of address, the KIMSI has 7 ports at F200, F400, ... ,FE00. It would seem more logical to put the I/O in page 21 or there abouts and gating the lower 8 bits onto the upper 8. This way any I/O board would work and some use would be made of that hole in the KIM memory map. The KIMSI is still a very good deal and I recomend it to anyone interested in cheep, S-100 memory, I/O etc.

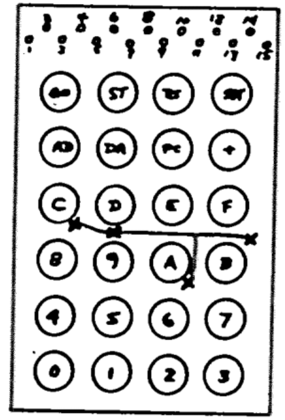
Included with the KIMSI was a note proposing KIMSI Notes. They hope to get enough material together about the KIMSI to fill a newsletter. I think that they should have given you a try and anounced the new Notes after they had the materal. Besides, they want another $6.

While I was waiting for the KIMSI, I was using a nice 8k board hooked directly to the KIM. This requires no permanent change to either board.

1. Connect the KIM address bus to the S-100 bus.
2. Connect the KIM data bus to the S-100 data in and out bus.
3. Connect RAM R/W (EZ) to pin 8 on IC 78.
4. Connect R/W (EV) to S-100 pin 47.
5. Connect DECODE ENABLE (AK) to pin 5 on IC 75.
6. Remove IC 74 and bend pin 4 out. Replace it so that pin 4 doesn't touch anything.

I wired steps 3 and 5 through unused pins on the S-100 connector. It worked fine with 6 inches of ribbon cable. Perhaps I should mention the board I used. It was the LOGOS-1 from Advanced Microcomputer Products and cost $219. When I got the KIMSI I removed the two jumpers and straightened the IC pin and it worked just fine..."

HAVING BOUNCY KEY PROBLEMS with your 'old' style keyboard? You'll be interested in this fixit from ROBERT DAHLSTROM, Harry Diamond Labs, 2800 Powder Mill Rd., Adelphi, MD 20783. This works!



KIM-1 KEYBOARD MODIFICATION

The keyboard on my KIM-1 had the "bouncy" key problem mentioned in User Notes #6. The problem is due to the use of the outer edge of the snap-action discs to jump over the center contact line on the keyboard pc. Since the discs are only held against the pc board with tape, the contact is poor. There are five of these jump-overs in series for the "C" key (four for the "9" key) thereby compounding the problem. To check for the problem, measure the resistance from keyboard pin 3 to pin 15 (numbered from left to right as shown) with the "C" key depressed. It should be less than about 10 ohms.

Fortunately, this problem can be easily corrected. My solution was to solder a thin wire jumper across these poor contacts as follows. Disassemble the keyboard by first removing the four screws on the back of the keyboard at the corners. Then remove the two remaining screws that hold the keyboard to the KIM-1 (note for reassembly that they are longer) being careful not to pull the keyboard pc board away from the KIM-1 board--it's only hanging by the solder at one end. With the KIM-1 up-side-down, separate the black keyboard panel from the keyboard pc board. (Mine snapped off suddenly when gently pried with my fingernail--then I picked up the keys from the floor). After cutting four small holes through the clear tape at the locations indicated by an X in the figure, the lines from "C" to "9", "D" to "9", "A" to "7" and the line to "B" are exposed. Connecting these points by soldering a thin wire between them routed as shown is sufficient to bridge the five potentially poor contacts. Good luck!

HERE'S AN IDEA FROM LEW EDWARDS (NJ)

A tip on using SST function to check out branches. Key FF into 00F1, then test all the BCS, BEQ, BMI &BVS branches. Next key in 00 and check out all the BCC, BNE, BPL & BVC branches. Seems obvious, but if you are like me it might not occur to you.

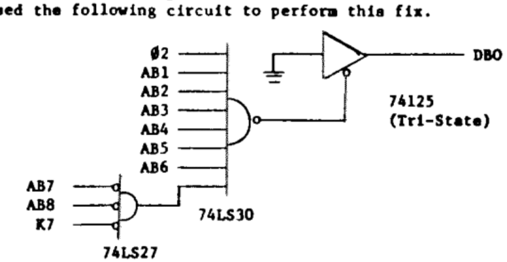
Here's a cure for a KIM problem you may may not have even known about from George Wells and Alex Engel at Jet Propulsion Laboratory, 4800 Oak Grove Dr., Pasadena, CA 91103......

A bug appears in the TTY software of both KIM and TIM which makes it difficult or impossible for either of these devices to receive TTY data at the maximum character rate for any baud rate other than 110 baud. For example, a paper type loader running at 10 cps (110 baud) will load correctly into KIM but at 30 cps (300 baud) a cross assembler on another computer has trouble loading the op codes into KIM.

The problem stems from the fact that there are two stop bits required for each character at 110 baud but only one stop bit for all other baud rates; and KIM and TIM were both written with the assumption that there will always be two stop bits per character.

Take a look at the "GETCH" (Get Character) subroutine located at 1E5A in KIM and you will see that it calls the 1 bit delay subroutine (JSR DELAY) 9 times and the half-bit delay subroutine (JSR DEHALF) twice for a total of 10 bits of delay. At 110 baud, since there is an extra stop-bit, KIM has at least 9 milliseconds to process the character; but at any other baud rate, KIM has no margins and may eventually lose sync depending on the length of the message, the baud rate, the baud rate drift, the character rate, and other factors which commonly come under the classifications of "gremlins", "noise", or "bad days".

If this sounds like a familiar problem to you and you're not satisfied with changing the TTY DELAY values at addresses 17F2, 3 (see issue #6, page 8 and 11) try this solution. It would be nice to fix KIM by eliminating the offending JSR DEHALF at address 1E7E. But since we can't do that, we'll do the next best thing which is to change it from a JSR DEHALF to a JSR DEHALF-1 which gives an immediate return from the subroutine. Note that DEHALF is located at 1EEB and at DEHALF-1 (1EEA) there is an RTS from the end of the previous routine. All we need to do is add some hardware to KIM to decode the second byte of the JSR DEHALF instruction and jam the LSB of the data bus to zero at that time. We have used the following circuit to perform this fix.



As mentioned before, TIM has the same problem except that it has a total delay of 10½ bits. However since we are unfamiliar with the operation of TIM we have not tried to fix it.

USER NOTES DEBUG SECTION

*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****

Issue 7 & 8, page 16-----pin 14 of the 74193 counters should go ground rather than Vcc.

Issue 7 & 8, page 2-----column 2,line 37 should read "To do this, set $039C to $01".

*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****DEBUG*****

SOME CORRECTIONS FOR THE TVT-6 CIRCUIT

The first comment comes from David Byrd, State Tech. Inst., 5983 Macon Cove, Memphis Tenn 38134

We just interfaced one of PAIA Electronics' TVT-6 video display kits (upper case letters only) to a Kim. While following Popular Electronic's debugging instructions, we noticed that our video monitor was displaying letters which were not complete because they were crowded together. Signal tracing turned up the fact that the LOAD signal was okay but the CLOCK signal presented only 3 cyles per microsecond instead of the specified 6 cycles. I tried replacing C5 (2200 pF) in the clock circuit with a smaller cap. The display looked better but it still needed improvement. After some "cut and try" we ended up with a 390 pF cap and a perfect video display.

Anyone who runs into a similar problem with one of these video display units might want to take note of our experience.

Also from Cass Lewart (12 Georjean Dr., Holmdel, NJ 07733)
"....I have built Don Lancaster's TVT. It works perfectly except that I changed C5 to 62 pf. and R11 to a 500 ohm pot. You may want to mention that we noticed a missing step in our program MINI DIS (First Book Of Kim). Step #364 should be 68 PLA.".....
Mr. Lewart also mentioned that he would be interested in setting up a program exchange for TVT programs. All you TVT-6 users should get in touch with him if you are interested.

From: Tim Bennett, 309 Mary St, Westerville, Ohio 43081

DOUBLE YOUR RAM. ADD 1 K, ON-BOARD,TO YOUR KIM-1.

All decoading and buffering is already available on your standard KIM-1 except that "K1" must be ORed with "K0" to enable inverter U16 pin 1. This requires 2 etch cuts, the addition of 2 diodes, 2 resistors, and a jumper along with 8 21L02 ram chips.

The 8 rams will be paralleled with your existing 6102 rams (U5-U12) except for pin 13 (Chip Enable). They could be soldered piggyback directly to the 6102's, however I was afraid this might cause overheating during operation. I chose to use sockets to lift my new rams from the existing to allow for air circulation. Normal chip DIP sockets are too bulky to permit soldering, thus Molex break-away connectors were used and they were perfect for this application.

Some special soldering techniques are required for a neat job on the RAMs. A 16 pin header or DIP socket (not the wire wrap kind) is used as a guide and holder for the molex connectors while soldering. Slip an 8 pin Molex section on each side of the socket with the break-away strip to the outside. now tin each of the Molex pins with a little solder where contact will be made with existing RAMs, leaving a tail of solder on the outside of the pins.

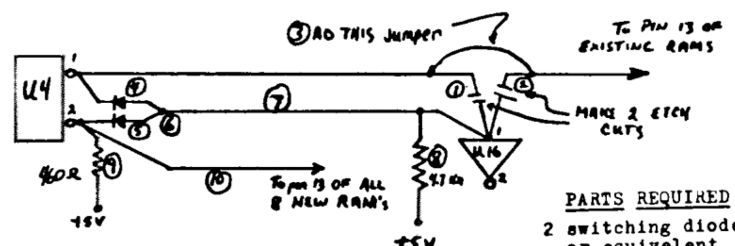
Dab a little soldering paste on each of the pins of the existing RAMs where contact will be made. Fit your socket assembly over an existing RAM. NOTE: don't solder pin 13 in the following step. If your assembly was properly prepared, a quick touch with an UNGAR PRINCESS iron will make a secure connection of each pin. Solder each pin (except pin 13) in this manner. Soldering will be easier if the chisel tip is bent to 45°. Carefully unplug the the guide and detatch the break-away strips by twisting back and forth at the scribe mark. Insert a 21L02 in your new socket keeping pin regisration the same as the origional 6102. Repeat this procedure for the remaining 7 RAMs. Varify that pin 13 of the 21L02's do not make contact with the 6102's.

Now implement the following changes to your "chip select" logic:

1. Cut etch at pin 1 of U16 on component side of pcb.
2. Cut etch at pin 1 of U16 on back side of pcb.
3. Jumper pin 1 of U4 (KØ) to pin 13 of U5.
4. Solder cathode (the end with the band) of one of your diodes to pin 1 of u4.
5. Solder cathode of other diode to pin 2 of U4.
6. Connect the anode end of the two diodes together.
7. Wire the anode end of the two diodes to pin 1 of U16
8. Connect a 4.7KΩ resistor from the anode of the diodes to a +5V etch.
9. Connect a 460Ω resistor from pin 2 of U4 to +5V.
10. Jumper pin 2 of U4 (K1) to pin 13 of all 8 21L02's
11. I brought +5V and GROUND in through both the application and the expansion connectors to carry the extra load.

The address of your second K of ram will be from 0400 to 07ff

I happen to have a supply of Molex strips. For a SASE and $2.00 I'll send enough for this modification + a few extra. Mail to Tim Bennett, 309 Mary st. Westerville, Ohio 43081.



TOOLS REQUIRED

UNGAR PRINCESS soldering iron
Soldering paste
A very steady hand
Solder
1 DIP SOCKET, solder tail or 16 PIN HEADER

PARTS REQUIRED

2 switching diodes (2N914) or equivelent
1 460 Ω, ¼ w. resistor
1 4.7 KΩ ¼ w. resistor
16 8 pin Molex sockets on break-away sockets P/N 05-30-0008
4ft 30 gauge wire
8 21L02 RAMS

Some comments and corrections from John P. Oliver (Dept of Physics, University of Florida, Gainesville, Florida 32611)

I have some comments and three corrections for my SUPERDUMP/LOAD routines published in Issue 7/8. a) Following the comment by James Davis in KUN #4, I have found that setting NPUL=$03 and TIMG+1=$02 greatly improves the reliability. I have had 100% success on Radio Shack SuperTape certified using Marchants routines from KUN#6. b) The program listing sent to you left out transmission of an EOT character. The instructions LDA #$04 , JSR OUTCHT should be inserted after the JSR OUTBT at $016A. This insertion unfortunately changes all the subroutine entry addresses. I will send a complete corrected listing to anyone who sends me a stamped, legal sized envelope. Without the EOT,SUPERLOAD sometimes will not return until the recorder is manually stopped. c) Most users will have recognized that the opcode 60 should be entered at $029D corresponding to the JSR instruction. My current version has the the following code at the end:

```
F0 04          BEQ EXIT
C6 CB  ERROR   DEC LFLG
C6 CB  ERROR2  DEC LFLG
A5 CB  EXIT    LDA LFLG
60             RTS
```

This addition results in the error flag being returned in the accumulator as well as being left at LFLG. Please note. SUPERDUMP/LOAD do not save the A,X,Y registers and the user is responsible for being sure that his flank is protected. This is not the best programming practice but I was trying for minimum subroutine length. I now have these routines in a more proper form stored in a 2708 EPROM which I have mounted on the KIM-1 board. The address lines are paralleled with those of the 6102's, the data lines are paralled with the DATAOUT lines of the 6102's. No buffering is needed. I had to replace the inverter in the RAM data buffer enable with a 4-input NAND gate combining K0,K1,K2,K3. I have also 'piggy backed' a set of 2102's on top of the 6102's, daisy chaining the CE's to K1, paralleling all other leads. I am trying to write a short article on this and other modifications I have used on our KIM's to give us KIM-E's (KIM Enhanced). I am not prepared to enter into correspondance on these changes at this time as I am trying to get ready for a 3 month visit to Warsaw for research. I am enclosing listings of START/STOP/WAIT which operate a high current,transistor driven relay in the recorder to start and stop it under program control. WAIT gives a 0.50 second delay which is adequate for my recorder. I only switch the motor power, leaving the electronics on, otherwise more than one second was needed at startup while capacitors charged in the amplifier. Finally,BEEP operates a loudspeaker driven from bit 4 of PBD. Entered with $00 in the A reg. one gets a sliding tone similar to that used for Phaser operation in the APPLE II Star Trek, with $FF one gets an opposite slide.

;BEEP ROUTINES FOR SPICA2

```
LOC    OP OPND   VALU  STMT  SCURCE STMT

                       0002         NAM ;BEEP ROUTINES FOR SPICA2
                       0003  ; **** SUPER BEEP ROUTINE ****
                       0004  ;       '0' IN ACC GIVES 'PHASER', 'FF' IN ACC GIVES BEEP
                 1702  0005  PBD    EQU $1702          ;DATA REGISTER B
                 1703  0006  PBDD   EQU $1703          ;DATA DIRECTION REGISTER B
                 00FD  0007  TMPX   EQU $00FD          ;USED FOR TEMP STORE OF ACC
1200             0000  0008         ORG $1200
1200   85 FD     00FD  0009  BEEP   STA TMPX           ;SAVE ACC
1202   8A              0010         TXA                ;SAVE X
1203   48              0011         PHA
1204   98              0012         TYA                ;SAVE Y
1205   48              0013         PHA
1206   A9 10     0010  0014         LDA #$10           ;SET UP OUPUT PORT ...
1208   0D 0317   1703  0015         ORA PBDD           ;... WITHOUT CHANGING ...
120B   8D 0317   1703  0016         STA PBDD           ;... OTHER LINES
120E   A0 00     0000  0017         LDY #$00
1210   98              0018  BEEP1  TYA
1211   AA              0019         TAX
1212   24 FD     00FD  0020         BIT TMPX           ;'FF'?
1214   30 05     121B  0021         BMI BEEP3          ;YES
1216   E8              0022  BEEP2  INX
1217   D0 FD     1216  0023         BNE BEEP2
1219   F0 03     121E  0024         BEQ BEEP4
121B   CA              0025  BEEP3  DEX
121C   D0 FD     121B  0026         BNE BEEP3
121E   AD 0217   1702  0027  BEEP4  LDA PBD            ;INVERT OUTPUT BIT
1221   49 10     0010  0028         EOR #$10
1223   8D 0217   1702  0029         STA PBD
1226   88              0030         DEY
1227   D0 E7     1210  0031         BNE BEEP1
1229   68              0032         PLA
122A   A8              0033         TAY                ;RESTORE Y
122B   68              0034         PLA
122C   AA              0035         TAX                ;RESTORE X
122D   A5 FD     00FD  0036         LDA TMPX
122F   60              0037         RTS                ;RETURN
```

START/STOP/WAIT ROUTINE

```
LOC    OP OPND   VALU  STMT  SCURCE STMT

                       0002         NAM START/STOP/WAIT ROUTINE
                       0003  ; **** START/STOP/WAIT ROUTINES FOR MAG TAPE ****
                 1702  0004  PBD    EQU $1702          ;DATA REGISTER B
                 1703  0005  PBDD   EQU $1703          ;DATA DIRECTION REGISTER B
11C0             0000  0006         ORG $11C0
11C0   48              0007  START  PHA                ;SAVE ACC
11C1   AD 0317   1703  0008         LDA PBDD           ;SET UP OUTPUT FOR ...
11C4   09 20     0020  0009         ORA #$20           ;... MAG TAPE CONTROL
11C6   8D 0317   1703  0010         STA PBDD
11C9   AD 0217   1702  0011         LDA PBD            ;PUT '0' ON PORT ...
11CC   29 DF     00DF  0012         AND #$DF           ;... WITHOUT CHANGING ...
11CF   8D 0217   1702  0013         STA PBD            ;... OTHER LINES
11D1   68              0014         PLA                ;RESTORE ACC
11D2   D0 0C     11E0  0015         BNE WAIT           ;WAIT 0.500 SECONDS
11D4   F0 0A     11E0  0016         BEQ WAIT
11D6   48              0017  STOP   PHA                ;SAVE ACC
11D7   AD 0217   1702  0018         LDA PBD            ;PUT '1' ON PORT ...
11DA   09 20     0020  0019         ORA #$20           ;... WITHOUT CHANGING ...
11DC   8D 0217   1702  0020         STA PBD            ;... OTHER LINES
11DF   68              0021         PLA
11E0   48              0022  WAIT   PHA                ;SAVE ACC
11E1   8A              0023         TXA                ;SAVE X
11E2   48              0024         PHA
11E3   98              0025         TYA                ;SAVE Y
11E4   48              0026         PHA
11E5   A0 CB     00CB  0027         LDY #$CB           ;WAIT 195 *255 LOOPS
11E7   A2 00     0000  0028         LDX #$00
11E9   2A              0029  WAIT1  ROL
11EA   2A              0030         ROL
11EB   2A              0031         ROL
11EC   CA              0032         DEX
11ED   D0 FA     11E9  0033         BNE WAIT1
11EF   88              0034         DEY
11F0   D0 F7     11E9  0035         BNE WAIT1
11F2   68              0036         PLA
11F3   A8              0037         TAY                ;RESTORE Y
11F4   68              0038         PLA
11F5   AA              0039         TAX                ;RESTORE X
11F6   68              0040         PLA                ;RESTORE ACC
11F7   60              0041         RTS                ;RETURN
```

...A FEW MORE KIM DEALERS.............

COMPUTER MART OF PENNSYLVANIA----550 DE KALB PIKE, KING OF PRUSSIA PA. 19406 (215-265-2580)

FALK-BAKER ASSOCIATES---382 FRANKLIN AVE., NUTLEY, NJ 07110 (201-661-2430)

COMPUTER MART ALSO CARRIES KIMSI S-100 MOTHERBOARD ADAPTORS, 'XIM' MONITOR SOFTWARE, THE FIRST BOOK OF KIM, AND THE CASSETTE TAPE (THE FIRST TAPE OF KIM????). BESIDES, THEY'RE GOOD PEOPLE.

FALK-BAKER CARRIES THE COMPLETE LINE OF OFFICIAL KIM STUFF (FROM THE FACTORY) AND EVEN CPU'S, MEMORY CHIPS, I/O PARTS, MANUALS, ETC.... GET THEIR FLYER.

NEXT ISSUE I'M GOING TO REVIEW SEVERAL ITEMS WHICH WILL BE OF INTEREST TO YOU KIMMERS: THE "KIMSI" MOTHERBOARD, THE 'MICRO-ADE' ASSEMBLER FROM PETER JENNINGS, AND A FANTASTIC NEW BOOK WHICH WILL PROVE VERY NECESSARY TO THOSE OF YOU WISHING TO LEARN MACHINE LANGUAGE PROGRAMMING. THE TITLE IS 'PROGRAMMING A COMPUTER : 6502', ITS PUBLISHED BY ADDISON-WESLEY, AUTHORED BY CAXTON FOSTER AND SHOULD BE AVAILABLE SOON AT YOUR DEALERS. IT'S EXCELLENT!!!!!